Development of localized surface plasmon resonance biosensors for the detection of Brettanomyces bruxellensis in wine

From Bisimulation to Simulation

Coarsest Partition Problems

R. Gentilini, C. Piazza and A. Policriti
({gentilin,piazza,policrit}@dimi.uniud.it)
Dip. di Matematica e Informatica - Università di Udine
Via Le Scienze, 206 - 33100 Udine - Italy

Abstract. The notions of *bisimulation* and *simulation* are used for graph reduction and are widely employed in many areas: Modal Logic, Concurrency Theory, Set Theory, Formal Verification, etc. In particular, in the context of Formal Verification they are used to tackle the so-called *state-explosion problem*.

The faster algorithms to compute the maximum bisimulation on a given labelled graph are based on the, crucial, equivalence between maximum bisimulation and *relational coarsest partition problem*. As far as simulation is concerned many algorithms have been proposed which turn out to be relatively inexpensive in terms of either time or space. In this paper we first revisit the state-of-the-art about bisimulation and simulation, pointing out the analogies and differences between the two problems. Then, we propose a generalization of the relational coarsest partition problem which is equivalent to the simulation problem. Finally, we conclude presenting an algorithm which exploits such a characterization and improves on either time or space complexity with respect to previously proposed algorithms for simulation.

Keywords: bisimulation, simulation, partition refinement problems

Table of Contents

1. Introduction

The behavior of a system or of a set of programs implementing a collection of cooperating subunits is naturally modelled by structures whose nodes describe the *possible states* and arrows represent *actions*. Even though very old and very much studied, this simple idea, so familiar to computer scientists as well as logicians, is recurrently at the ground of important progresses in a variety of sub-fields of informatics.

As it is natural to expect, the main, somehow intrinsic, difficulties with this approach to modelling, are the intricacies buried in the modelling structures and the sheer size of the obtained structures. In fact, the principal purpose of the modelling activity is to be able to perform automatically, within such structures, the search for properties, specified in suitably designed languages, that can be proved or disproved, during the design phase. In this general framework, one of the main tools available to cope with the explosion of the size of the modelling structures (Kripke structures, automata, transition systems of various kind), is to use algorithms reducing the size of the structure before any "reasoning" takes place on them.

In this paper we consider systematically two of such possible reductions (*bisimulation* and *simulation*) from an algorithmic point of view. In particular, our purpose is to show how there is a *fil rouge* connecting such notions which consists in casting them into partition refinement problems (coarsest partition problems). Such a formulation is then showed to be engine for the design of fast and efficient algorithms along rather natural paths in both cases and, we hope, should also shed some light on the concrete nature of this kind of graph reduction.

The paper continues, after introducing some preliminary material, with a section dealing with the state of the art relative to the study of bisimulation and simulation, respectively, providing a specific preliminary discussion on some history on each of the two notions and the algorithmic tools used for their implementations. In the subsequent section we present a new result in the form of a novel algorithm for simulation, based on the mapping of the notion on a suitable coarsest partition problem.

Preliminary versions of the results presented here were discussed in (DPP01) and (GPP02a).

2. Preliminaries

In this section we introduce the basic notations we use in the rest of the paper.

DEFINITION 2.1. *Let T be a set and $Q \subseteq T \times T$ a binary relation over T:*

- *Q is said to be a* quasi order *over T if and only if Q is reflexive and transitive;*
- *Q is said to be a* partial order *over T if and only if Q is reflexive, antisymmetric, and transitive;*
- *Q is said to be* acyclic *if and only if its transitive closure is antisymmetric.*

We will use Q^+ to refer to the transitive closure of Q and Q^* to refer to the reflexive and transitive closure of Q.

Notice that if a relation is acyclic, then it is antisymmetric, while the converse does not hold (unless it is transitive).

DEFINITION 2.2 (Labelled Graphs). *A triple $G = (N, \rightarrow, \Sigma)$ is said to be a* labelled graph *if and only if $G^- = (N, \rightarrow)$ is a finite graph and Σ is a partition over N. We say that two nodes $a, b \in N$ have the same* label *if they belong to the same class of Σ.*

An equivalent way to define *labelled graphs* is to use a labelling function $\ell : N \rightarrow L$, where L is a finite set of labels (inducing a partition Σ_L of N). Given a node $a \in N$ we will use $[a]_\Sigma$ (or $[a]$, if Σ is clear from the context) to denote the class of Σ to which a belongs.

EXAMPLE 2.3. *A* Kripke Structure *is a labelled graph and, vice-versa, each labelled graph can be seen as a Kripke Structure in which two worlds satisfy the same set of atomic propositions if and only if their labels are equal.*

DEFINITION 2.4 (Bisimulation). *Let $G = (N, \rightarrow, \Sigma)$ be a direct labelled graph. A relation $\asymp \subseteq N \times N$ is said to be a* bisimulation *over G if and only if:*

1. $a \asymp b \Rightarrow [a]_\Sigma = [b]_\Sigma$;
2. $(a \asymp b \wedge a \rightarrow c) \Rightarrow \exists d(c \asymp d \wedge b \rightarrow d)$;
3. $(a \asymp b \wedge b \rightarrow d) \Rightarrow \exists c(c \asymp d \wedge a \rightarrow c)$.

We say that a and b are bisimilar *($a \equiv_b b$) if there exists a bisimulation $\asymp$ such that $a \asymp b$.*

Notice that a bisimulation can be neither reflexive nor transitive, and not even symmetric, however the reader can easily verify that given an arbitrary bisimulation its reflexive, symmetric, and transitive closure is always a bisimulation. The proof of the following lemma can be found in (Mil80).

LEMMA 2.5. *Let $G = (N, \rightarrow, \Sigma)$ be a labelled graph. The relation $\equiv_b$ is an equivalence relation over N and a bisimulation over G. Moreover, $\equiv_b$ is the maximum bisimulation, i.e. if $\asymp$ is a bisimulation over G, then $\asymp \subseteq \equiv_b$.*

Since $\equiv_b$ is an equivalence relation, it is possible to consider the quotient $B = N/\equiv_b$. We will use the notation $[a]_b$ to denote the equivalence class to which a belongs in B.

DEFINITION 2.6 (Bisimulation Problem). *Given a labelled graph $G = (N, \rightarrow, \Sigma)$ the* bisimulation problem *over G consists in computing the quotient $B = N/\equiv_b$, where $\equiv_b$ is the maximum bisimulation over G.*

DEFINITION 2.7 (Simulation). *Let $G = (N, \rightarrow, \Sigma)$ be a labelled graph. A relation $\leq \subseteq N \times N$ is said to be a* simulation *over G if and only if:*

1. $a \leq b \Rightarrow [a]_\Sigma = [b]_\Sigma$;
2. $(a \leq b \wedge a \rightarrow c) \Rightarrow \exists d(c \leq d \wedge b \rightarrow d)$.

In this case we also say that b simulates *a.*
We say that b and a are sim-equivalent *($b \equiv_s a$) if there exist two simulations $\leq_1$ and $\leq_2$, such that $a \leq_1 b$ and $b \leq_2 a$.*

Again, a simulation can be neither reflexive nor transitive (e.g. the empty relation is always a simulation), but given an arbitrary simulation its reflexive and transitive closure is always a simulation.

A simulation $\leq_s$ over G is said to be *maximal* if for all the simulations $\leq$ over G it holds $\leq \subseteq \leq_s$.

LEMMA 2.8. *Given a labelled graph $G = (N, \rightarrow, \Sigma)$ there always exists a unique maximal simulation $\leq_s$ over G. Moreover $\leq_s$ is a quasi order.*

COROLLARY 2.9. *Let $G = (N, \rightarrow, \Sigma)$ be a labelled graph. The relation $\equiv_s$ (sim-equivalence) is an equivalence relation over N.*

Proof. Let $\leq_s$ be the maximal simulation relation over G. From the definition of sim-equivalence we have that

$$a \equiv_s b \text{ iff } a \leq_s b \wedge b \leq_s a.$$

Since $\leq_s$ is a quasi order, $\equiv_s$ is an equivalence relation. □

Also in the case of simulation, since $\equiv_s$ is an equivalence relation, it is possible to consider the quotient $S = N/\equiv_s$. We will use the notation $[a]_s$ to denote the equivalence class to which a belongs in S.

DEFINITION 2.10 (Simulation Problem). *Given a labelled graph* $G = (N, \rightarrow, \Sigma)$ *the* simulation problem *over* G *consists in computing the quotient* $S = N/\equiv_s$, *where* $\equiv_s$ *is the sim-equivalence over* G.

3. State-of-the-art

3.1. Bisimulation

The notion of bisimulation, formally defined in Section 2, has been introduced with different purposes in many areas related to Computer Science. In Modal Logic it was introduced by van Benthem (cf. (Ben76)) as an equivalence principle between Kripke structures. In Concurrency Theory it was introduced by Park (cf. (Par81)) for testing observational equivalence of the Calculus of Communicating Systems (CCS). In Set Theory, it was introduced by Forti and Honsell (cf. (FH83)) as a natural principle replacing extensionality in the context of non well-founded sets. As far as Formal Verification is concerned (cf. (CGP99)), several existing verification tools make use of bisimulation in order to minimize the state spaces of systems description ((CS96; Bou98; FGK$^+$96; Ros94)), since bisimulation preserves the whole μ-calculus. The reduction of the number of states is important both in compositional and in non-compositional Model Checking. Bisimulation serves also as a mean of checking equivalence between transition systems. In the context of Security many non interference properties are based on checking bisimulation between systems (cf. (FG97)).

From a computational point of view, the main reason for the fortune of bisimulation and for its fast solution lie in the equivalence between the bisimulation problem and the problem of determining the *coarsest partition* of a set *stable* with respect to a given relation.

DEFINITION 3.1 (Stability). *Let* $\rightarrow$ *be a binary relation on the set* N, $\rightarrow^{-1}$ *its inverse relation, and* Σ *a partition of* N. Σ *is said to be* stable *with respect to* $\rightarrow$ *iff for each pair* α, γ *of blocks of* Σ, *either* $\alpha \subseteq \rightarrow^{-1}(\gamma)$ *or* $\alpha \cap \rightarrow^{-1}(\gamma) = \emptyset$.

We say that a partition Π *refines* a partition Σ (Π is *finer* than Σ) if each block of Π is included in a block of Σ.

DEFINITION 3.2 (Coarsest Stable Partition Problem). *Let $\rightarrow$ be a binary relation on the set N and Σ a partition over N. The* coarsest stable partition problem *is the problem of finding the coarsest partition B refining Σ that is stable with respect to $\rightarrow$.*

This problem, that emerged also naturally in automata minimization, is equivalent to the bisimulation problem.

PROPOSITION 3.3. *Let $G = (N, \rightarrow, \Sigma)$ be a labelled graph.*

(*i*) *Let Π be a partition over N refining Σ and stable with respect to $\rightarrow$. Then $\asymp_\Pi$ defined as:*

$$a \asymp_\Pi b \text{ iff } \exists \alpha \in \Pi\, (a \in \alpha \wedge b \in \alpha)$$

is a bisimulation over G.

(*ii*) *Let $\asymp$ be a bisimulation over G which is also an equivalence relation. Then $\Pi_\asymp = \{[a]_\asymp : a \in N\}$, where $[a]_\asymp = \{b \in N : a \asymp b\}$, is a partition stable with respect to $\rightarrow$.*

Next corollary follows immediately from the previous proposition.

COROLLARY 3.4. *Let $G = (N, \rightarrow, \Sigma)$ be a graph. Computing the maximum bisimulation $\equiv_b$ on G or finding the coarsest stable partition of N refining Σ and stable with respect to $\rightarrow$ are equivalent problems.*

The first significant result related to the algorithmic solution of the bisimulation problem is in (Hop71), where Hopcroft presents an algorithm for the minimization of the number of states in a given finite state automaton. Hopcroft's result has been subsequently clarified and improved in (Gri73; Knu01) The problem is equivalent to that of determining the coarsest partition of a set *stable* with respect to a finite set of functions. A variant of this problem is studied in (PTB85), where it is shown how to solve it in linear time in case of a single function. Finally, in (PT87) Paige and Tarjan solved the problem for the general case (which is the same as solving the bisimulation problem) in which the stability requirement is relative to a relation $\rightarrow$ (on a set N) with an algorithm whose complexity is $O(|\rightarrow| \log |N|)$. The main feature of the linear solution to the single function coarsest partition problem (cf. (PTB85)), is the use of a *positive* strategy in the search for the coarsest partition: the starting partition is the partition with singleton classes and the output is built via a sequence of steps in which two or more classes are merged. Instead, Hopcroft's solution to

the (more difficult) many functions coarsest partition problem is based on a (somehow more natural) *negative* strategy: the starting partition is the input partition and each step consists of the split of all those classes for which the stability constraint is not satisfied. The interesting feature of Hopcroft's algorithm lies in its use of a clever ordering (the so-called "process the smallest half" ordering) for processing classes that must be used in a split step. Starting from an adaptation of Hopcroft's idea to the relational coarsest partition problem, Paige and Tarjan succeeded in obtaining their fast solution (PT87). The process the smallest half policy establishes that if a block α' is split into α and $\alpha' \setminus \alpha$, and α has less elements than $\alpha' \setminus \alpha$, then we can use (only) α as splitter and ignore $\alpha' \setminus \alpha$. In their generalization of this policy Paige and Tarjan determine how to perform $|\alpha|$ steps also when it is necessary to use both α and $\alpha' \setminus \alpha$ as splitters.

In (KS90) Kannellakis and Smolka notice that the algorithm by Paige and Tarjan (PT87) for the relational coarsest partition problem can be used to determine the maximum bisimulation over a graph $G = (N, \rightarrow, \Sigma)$.

In (BFH90) Bouajjani, Fernandez, and Halbwachs propose an algorithm for the relational coarsest partition problem tailored for the context of on-the-fly Model Checking, i.e. they stabilize only reachable blocks. In (LY92) Lee and Yannakakis improve this method by using only reachable blocks to stabilize the reachable blocks.

From a more abstract point of view, an interesting property of the notion of bisimulation is that the bisimulation problem over labelled graphs is equivalent the one over unlabelled ones. Given a labelled graph $G = (N, \rightarrow, \Sigma)$ it is possible to build a graph $G' = (N', \rightarrow')$, with $N \subseteq N'$ and $\rightarrow \subseteq \rightarrow'$, such that for all $a, b \in N$ $a \equiv_b b$ over G if and only if $a \equiv_b b$ over G' (see (DPP01; Pia02)). As a matter of fact, in the context of non-well-founded sets graphs are used to represent sets (see (Acz88)), hence they have no labels and the notion of bisimulation determines set-equalities. The fact that the problem with labels can be reduced to the one without labels means that the set-theoretic formulation of the bisimulation problem is general enough to embed all the other bisimulation problems.

In (DPP01) exploiting the set-theoretic formulation of the bisimulation problem an algorithm which optimizes the algorithm in (PT87) is presented. The worst case complexity of the algorithm described in (DPP01) is equal to the worst case complexity of the algorithm proposed in (PT87), but in a large number of cases it obtain better performances. In particular, this algorithm integrates positive and negative strategies and the combined strategy is driven by the set-theoretic

notion of *rank* of a set. In the case of acyclic graphs the rank of a node a is nothing but the length of the longest path from a to a leaf.

DEFINITION 3.5. *Let $G = (N, \rightarrow)$ be an acyclic graph. The* rank *of a node a is recursively defined as follows:*

$$\begin{cases} rank(a) = 0 & \text{if } a \text{ is a leaf} \\ rank(a) = 1 + \max\{rank(c) : a \rightarrow c\} & \text{otherwise} \end{cases}$$

In the general case the rank of a node a is the length of the longest path from a node c, reachable from a, to a leaf such that all the nodes involved in the path do not reach cycles. In order to give a formal definition we first introduce the notion of graph of the strongly connected components.

DEFINITION 3.6 (Strongly Connected Components). *Given a graph $G = (N, \rightarrow)$, let $G^{scc} = (N^{scc}, \rightarrow^{scc})$ be the graph obtained as follows:*

$$\begin{aligned} N^{scc} &= \{C : C \text{ is a strongly connected component in } G\} \\ \rightarrow^{scc} &= \{\langle C(a), C(c)\rangle : C(a) \neq C(c) \text{ and } a \rightarrow c\} \end{aligned}$$

Given a node $a \in N$, we refer to the node of G^{scc} associated to the strongly connected component of a as $C(a)$.

Observe that G^{scc} is acyclic.

We also need to distinguish between the well-founded part and the non-well-founded part of a graph G.

DEFINITION 3.7 (Well-Founded Part). *Let $G = (N, \rightarrow)$ and $a \in N$. $G(a) = (N(a), \rightarrow\restriction N(a))$ is the subgraph[1] of G of the nodes reachable from a. $WF(G)$, the well-founded part of G, is $WF(G) = \{a \in N : G(a)$ is acyclic$\}$.*

DEFINITION 3.8 (Rank). *Let $G = (N, \rightarrow)$. The* rank *of a node a of G is defined as:*

$$\begin{cases} rank(a) = 0 & \text{if } a \text{ is a leaf in } G \\ rank(a) = -\infty & \text{if } C(a) \text{ is a leaf in } G^{scc} \text{ and } a \text{ is not a leaf in } G \\ rank(a) = \max(\{1 + rank(c) : C(a) \rightarrow^{scc} C(c), c \in WF(G)\} \cup & \\ \qquad \{rank(c) : C(a) \rightarrow^{scc} C(c), c \notin WF(G)\}) & \text{otherwise} \end{cases}$$

Two important properties of the notions of rank, proved in (DPP01), suggest to exploit it in the bisimulation problem:

- if $a \equiv_b b$, then $rank(a) = rank(b)$;

[1] We use $\rightarrow\restriction N(a)$ to denote the restriction of $\rightarrow$ to $N(a)$.

– if $\equiv_b$ has been computed on the nodes of rank less than i, then it is possible to compute $\equiv_b$ on the nodes at rank i.

Moreover, given a graph $G = (N, \rightarrow)$ the ranks can be assigned to the nodes of G in time $O(|N| + | \rightarrow |)$ using the Tarjan's algorithm for the strongly connected components. This means that using the ranks inside a bisimulation algorithm does not increase its asymptotic time complexity.

The algorithm in (DPP01) first splits the graph into ranks, then uses (PTB85) and (PT87) as subroutines in increasing order of ranks. It terminates in linear time in many cases, for example when the input problem corresponds to a bisimulation problem on acyclic models. It operates in linear time in other cases as well and, in any case, it runs at a complexity less than or equal to that of the algorithm by Paige and Tarjan (PT87). Moreover, the partition imposed by the rank allows to process the input without storing the entire structure in memory at the same time. This allows (potentially) to deal with largest graphs than those treatable using a Paige and Tarjan-like approach. A symbolic version of the algorithm in (DPP01) has been presented in (DGPP02), where it has also been proposed the use the notion of rank inside Model Checking procedures (i.e., not only to compute the bisimulation quotient, but to evaluate the CTL formulae).

In (PP01) the bisimulation problem is tackled again from a set-theoretic point of view. In particular, it is shown how in the case of acyclic graphs a compact version of the Ackermann's encoding can be used to solve the bisimulation problem, then the encoding is extended to the general case.

In Section 2 we presented the bisimulation problem over a labelled graph G. It is possible to formulate the problem using two graphs G_1 and G_2 for which it is convinient to define *labelled graphs* using a labelling function $\ell : N \rightarrow L$, where L is a finite set of labels common for all the graphs.

DEFINITION 3.9 (Two graphs Bisimulation). *Let $G_1 = (N_1, \rightarrow_1, \ell_1)$ and $G_2 = (N_2, \rightarrow_2, \ell_2)$ be two labelled graphs. A relation $\asymp \subseteq N_1 \times N_2$ is said to be a* bisimulation *between G_1 and G_2 if and only if:*

1. $a \asymp b \Rightarrow \ell_1(a) = \ell_2(b)$;
2. $(a \asymp b \wedge a \rightarrow c) \Rightarrow \exists d(c \asymp d \wedge b \rightarrow d)$;
3. $(a \asymp b \wedge b \rightarrow d) \Rightarrow \exists c(c \asymp d \wedge a \rightarrow c)$;
4. *for each $a \in N_1$ there exists $b \in N_2$ such that $a \asymp b$;*

5. for each $b \in N_2$ there exists $a \in N_1$ such that $a \asymp b$;

Two graphs G_1 and G_2 are said to be bisimilar *if there exists a bisimulation between G_1 and G_2.*

This formulation of the problem allows us to use bisimulation as a reduction procedure quotienting G onto the domain $N/\equiv_b$. Such a reduced graph can be build thanks to an important property of $\equiv_b$ (and of all the bisimulations that are equivalence relations):

$$\forall a, b, c((a \to c \wedge b \in [a]_b) \Rightarrow \exists d(d \in [c]_b \wedge b \to d)) \quad (\flat)$$

Hence, we can define the quotient graph $G/\equiv_b = (N_{\equiv_b}, \to_{\equiv_b}, \ell_{\equiv_b})$ as:

$$\begin{aligned} N_{\equiv_b} &= N/\equiv_b \\ [a]_b \to_{\equiv_b} [c]_b &\Leftrightarrow \exists c_1(c_1 \in [c]_b \wedge a \to c_1) \\ \ell_{\equiv_b}([a]_b) &= \ell(a). \end{aligned}$$

PROPOSITION 3.10. *Let $G = (N, \to, \ell)$ be a labelled graph. The graph $G/\equiv_b$ is the minimum graph bisimilar to G.*

3.2. Simulation

The notion of simulation, introduced in Section 2, first defined by Milner in (Mil71) as a means to compare programs, is very similar (less demanding, in fact) to the notion of bisimulation. Since the conditions in the definition of simulation are weaker than the ones in the definition of bisimulation, simulation provides, for example in the context of Formal Verification, a better space reduction than bisimulation. Nevertheless simulation is still adequate for the verification of all the formulae of the branching temporal logic without quantifiers switches (DGG93) (e.g. the formulae of ACTL*, see (GL94)). As explained in (HHK95) "in many cases, neither trace equivalence nor bisimilarity, but similarity is the appropriate abstraction for computer-aided verification ...". In the case of finite-state systems the similarity quotients can be computed in polynomial time, while this is not the case for trace equivalence quotients. In the case of infinite-state systems, finitely represented using hybrid automaton and other formalisms, the similarity quotients can be computed symbolically and in many cases the quotients are finite (see (HHK95)).

Several polynomial-time algorithms to solve the simulation problem on finite graphs have been proposed: the ones in (Blo89), (CPS93), and (CS92) achieve $O(|N|^6| \to |)$, $O(|N|^4| \to |)$, and $O(| \to |^2)$ time complexities, respectively. A simulation procedure running in $O(|N|| \to |)$

time was independently discovered in (HHK95) and (BP95). All of the algorithms just mentioned ((Blo89), (CPS93), (CS92), (BP95), (HHK95)) obtain the similarity quotient $S = N/\equiv_s$ as a by-product of the computation of the entire similarity relation $\leq_s$ on the set of states N. Their space complexity is then limited from below by $O(|N|^2)$.

Recently Bustan and Grumberg in (BG00) and Cleaveland and Tan in (CT01) improved the above results.

The procedure by Bustan and Grumberg (BG00) gives in output the quotient structure with respect to $\equiv_s$ and the simulation relation among the classes of $S = N/_{\equiv_s}$ without computing the entire simulation on N. Hence, its space requirements (often more critical, especially in the field of verification) depends on the size of S and are lower than the ones of the algorithms in (Blo89), (CPS93), (CS92), (BP95), and (HHK95). In more detail, the algorithm described in (BG00) uses only $O(|S|^2+|N|\log(|S|))$ space whereas its time complexity is rather heavy: it is $O(|S|^4(|\rightarrow| + |S|^2) + |S|^2|N|(|N| + |S|^2))$.

The procedure in (CT01) combines the fix-point calculation techniques in (BP95) and (HHK95) with the bisimulation-minimization algorithm in (PT87). A system, G_2, is determined being or not capable of simulating the behavior of G_1, by interleaving the minimization via bisimulation of the two systems with the computation of the set of classes in G_2 able to simulate each class in G_1. The time complexity achieved is $O(|B_1||B_2| + |\rightarrow_1|\log(|N_1|) + |B_1||\rightarrow_2| + |\varepsilon_1||B_2|)$, where ε_i and B_i represent the bisimulation reduced relation and state-space of T_i. Compared with the time complexities of (HHK95) and (BP95), the latter expression has many occurrences of $|N_i|$ and $|\rightarrow_i|$ replaced with $|B_i|$ and $|\varepsilon_i|$. Indeed, the experimental results in (CT01) prove that the procedure by Cleaveland and Tan outperform the ones in (HHK95) and (BP95). The space complexity of (CT01) depends on the product of the sizes of the two bisimulation quotients involved. Being bisimulation finer than simulation, such a space requirement may be more demanding than the one in (BG00).

As in the case of bisimulation, it is possible to formulate the simulation problem between two graphs. In particular, a "two graphs" formulation is used in (CT01).

DEFINITION 3.11 (Two graphs Simulation). *Let* $G_1 = (N_1, \rightarrow_1, \ell_1)$ *and* $G_2 = (N_2, \rightarrow_2, \ell_2)$ *be two labelled graphs. A relation* $\leq \subseteq N_1 \times N_2$ *is said to be a* simulation *from* G_1 *to* G_2 *if and only if:*

1. $a \leq b \Rightarrow \ell_1(a) = \ell_2(b)$;
2. $(a \leq b \wedge a \rightarrow c) \Rightarrow \exists d(c \leq d \wedge b \rightarrow d)$;

3. for each $a \in N_1$ there exists $b \in N_2$ such that $a \leq b$;

A graph G_1 is simulated *by a graph G_2 (G_2 simulates G_1) if there exists a simulation from G_1 to G_2. Two graphs G_1 and G_2 are sim-equivalent if both G_1 simulates G_2 and G_2 simulates G_1*

Hence, given a graph G it is possible to consider the problem of determining the minimum graph sim-equivalent to G. In the context of simulation minimality is measured in terms of both number of nodes and edges. In (BG00) it has been proved that there always exists a unique smallest labelled graph that is sim-equivalent to G, i.e. there is a unique way to put a minimum number of edges between the elements of $S = N/\equiv_s$ in order to obtain a labelled graph sim-equivalent to G. In the case of bisimulation this was a trivial consequence of the property ($\flat$) (see Section 3.1), since if a node reaches with an edge an equivalence class, then all the nodes that are equivalent to it reach with an edge the same equivalence class. This is equivalent to say that

$$\forall a, c([a]_b \cap \rightarrow^{-1}([c]_b) \neq \emptyset \Rightarrow [a]_b \subseteq \rightarrow^{-1}([c]_b)),$$

which is at the basis of the characterization of the bisimulation problem as (relational) coarsest partition problem. In the case of simulation it is possible that there exists a node a which reaches with an edge a node c, while a node which is sim-equivalent to a does not reach any node sim-equivalent to c. When this happens we always have that there exists a node d such that a reaches d and $c \leq_s d$, and we say that c is a *little brother* of d. Hence, the property which holds for the maximum simulation is that

$$\forall a, b, c((a \rightarrow c \wedge b \in [a]_s) \Rightarrow \exists d(c \leq_s d \wedge b \rightarrow d)) \quad (\sharp)$$

which in particular, since we are working on finite graphs, implies that

$$\forall a, c([a]_s \cap \rightarrow^{-1}([c]_s) \neq \emptyset \Rightarrow \exists [d]_s([c]_s \leq_s [d]_s \wedge [a]_s \subseteq \rightarrow^{-1}([d]_s))).$$

This property, at the basis of the *generalized stability* condition that we will introduce in the second part of this paper, provides a characterization of the simulation problem in terms of partitioning problem. An immediate consequence is that there is a unique way to put a minimum number of edges among the elements of $N/\equiv_s$ in order to obtain a graph sim-equivalent to G: put an edge from $[a]_s$ to $[d]_s$ if and only if it holds $[a]_s \subseteq \rightarrow^{-1}([d]_s)$ and there is no class $[c]_s$ such that $[a]_s \cap \rightarrow^{-1}([c]_s) \neq \emptyset$ with $d \leq_s c$ (i.e. d is not a little brother).

EXAMPLE 3.12. *Consider the two graphs in Figure 1. The graph on the right is the quotient with respect to the maximum simulation of the*

graph on the left. In the graph on the left there are no sim-equivalent nodes, hence all the classes in the quotient structure are singletons. In the graph on the left there is a node with label B which is a little brother, hence in the quotient structure we delete an edge.

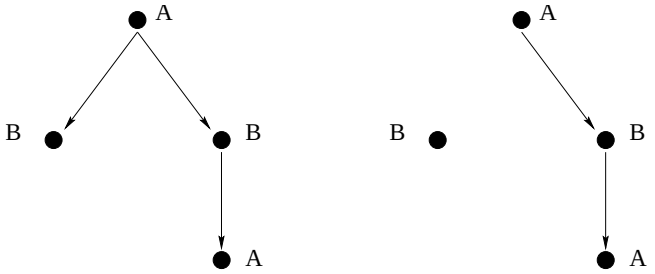

Figure 1. Example of little brother and of minimization in terns of edges.

Another important difference between bisimulation and simulation is that while in the case of bisimulation working without labels is not a restriction (see Section 3.1), in the case of simulation this is not the case. This is a consequence of the following result whose proof is left to the reader.

LEMMA 3.13. *Let $G = (N, \rightarrow, \Sigma)$ be a graph such that $\Sigma = \{N\}$, i.e. all the nodes have the same label. Given a node $a \in N$ let $G(a)$ be the subgraph of G of the nodes reachable from a.*

- *If $G(a)$ is acyclic, then for all $b \in N$*

$$b \leq_s a \qquad \textit{iff} \qquad G(b) \textit{ is acyclic and } \textit{rank}(b) \leq \textit{rank}(a).$$

- *If $G(a)$ is cyclic, then for all $b \in N$*

$$b \leq_s a.$$

This means that in the case without labels (i.e. only one label) all the nodes which belong to the non-well-founded part of the graph are sim-equivalent.

EXAMPLE 3.14. *Consider the graph in Figure 2. If we try to remove the labels A and B by adding new nodes and we do not want to use labels also on the new nodes, then, no matter how many new nodes and edges we add, the two nodes of the original graph become sim-equivalent.*

We anticipate here that in the second part of this paper we will introduce a "partial" unlabelling which pushes the labels down on new

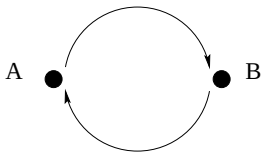

Figure 2. Example with Labels.

leaves. However, we introduce this unlabelling only to prove that the simulation problem is equivalent to a generalization of the coarsest partition problem. The unlabelling is not necessary in the algorithm we present.

The previous lemma has also strong implications on the use of the notion of rank in the computation of $\equiv_s$. It implies that

$$a \equiv_s b \;\not\Rightarrow\; rank(a) = rank(b),$$

since in a graph with only one label all the non-well-founded nodes are sim-equivalent even if they are at different ranks. Lemma 3.13 suggests that a reasonable definition of rank to be used in the computation of $\equiv_s$ could be

$$rank^*(a) \;=\; \begin{cases} \max(\{1 + rank(c) \mid C(a) \rightarrow^{scc} C(c)\} & \text{if } a \in WF(G) \\ +\infty & \text{otherwise} \end{cases}$$

i.e. on the well-founded part of the graph we use the usual notion of rank, while all the nodes in the non-well-founded part have rank $+\infty$. Using such a definition we obtain

$$a \equiv_s b \;\Rightarrow\; rank^*(a) = rank^*(b);$$
$$a \leq_s b \;\Rightarrow\; rank^*(a) \leq rank^*(b).$$

Unfortunately this implies that, whenever we use a technique which needs to compute $\leq_s$ in order to determine $\equiv_s$, when we process nodes at rank i we have to keep into account also all the nodes at rank less than i.

A second disadvantage of the use of $rank^*$ is that it is reasonable to believe that at least one half of the nodes have rank $+\infty$, which means that it does not split enough the computation.

For this last reason in (GPP02b) in order to exploit a notion of rank in the simulation computation the opposite approach is taken, i.e. a notion of rank which splits the graph as much as possible is introduced admitting to have sim-equivalent nodes with different ranks. In any case, what is required to the notion of rank is the property that the information at rank i is enough to compute the similarity relations

among the nodes of rank at most i. In particular, we want to avoid the situation in which a and b are two nodes whose rank is at most i, at the end of the i-th iteration we believe that $a \equiv_s b$ (resp. $a \not\equiv_s b$) and at the end of the $(i+k)$-th iteration we discover that $a \not\equiv_s b$ (resp. $a \equiv_s b$). A necessary and sufficient condition to obtain the above property is that:

if $a \to b$, then the rank of b is not greater than the rank of a.

The following notion of rank turns out to satisfy the above condition and splits the graph as much as possible:

DEFINITION 3.15 (sim-Rank). *Let $G = (N, \to, \Sigma)$, the $rank_s$ of a node is recursively defined as follows:*

$$rank_s(a) = \begin{cases} 0 & \text{if } a \text{ is a leaf in } G^{scc} \\ \max\{1 + rank_s(c) \mid C(a) \to^{scc} C(c)\} & \text{otherwise} \end{cases}$$

This means that at each rank a set of strongly connected components is considered and the order in which these sets of strongly connected components are considered is determined by the usual (well-founded) notion of rank on the graph of the strongly connected components.

3.3. Model Checking and Symbolic Algorithms

As already mentioned, one of the main application of the notions of bisimulation and simulation comes from the area of Formal Verification. The main disadvantage of Model Checking is the so-called *state-explosion* that can occur if the system being verified has many components that can make transitions in parallel. In this case the number of global system states may grow exponentially with the number of processes. Different techniques have been developed and are still under active investigation to solve this problem: OBDD and Symbolic Model Checking (McM93), Abstract Model Checking (DGG97), partial order reductions (CGP99), equivalence reductions (LY92; HHK95). Bisimulation and simulation belong to this last group of techniques, since they allow to obtain a smaller structure which satisfies the same properties of the input one (preservation of various logics). Bisimulation guarantees preservation of branching-time temporal logics such as CTL and CTL* (CE82), while simulation preserves of the universal fragment of these logics (ACTL and ACTL* (GL94)).

For this reason many Model Checking tools integrate subroutines to perform both bisimulation and simulation reductions (CS96; Bou98; FGK$^+$96; Ros94). The verification environment XEVE (Bou98) provides bisimulation tools which can be used for both minimization and equivalence test. The Concurrency Workbench (CWB) (CPS93) tests

bisimulation using techniques based on the Kanellakis and Smolka algorithm. The Compositional Security Checker (CoSec) (FG97) exploits the bisimulation algorithm implemented in CWB in order to test information flow security properties. In the Concurrency Workbench of the New Century (CWB-NC) (CS96) the underling bisimulation algorithm is the Paige and Tarjan one, while the simulation algorithms used are the ones presented in (BP95; CT01).

Sometimes different techniques are put together to obtain better reductions in the memory requirements. In particular, Symbolic Model Checking, which is based on the use of Ordered Binary Decision Diagrams (OBDDs) (see (Bry85; McM93)) to represent the input structure and sets of states, is sometimes associated to the use of bisimulation and simulation. Unfortunately, not all the algorithms fit with the use of OBDDs: the main advantage of OBDDs is that they compactly represent big sets of states, hence it is not reasonable to use algorithms which need to consider each state separately. A classical example is Tarjan's algorithm (or any version of it) for the strongly connected components, which is inapplicable on OBDD since it requires to manipulate each node singularly in order to assign it its finishing-time.

In the *symbolic* case, i.e. when OBDDs are used as basic data-structures, a popular bisimulation algorithm is (BdS92) by Bouali and de Simone. It implements the naïve negative strategy optimizing the boolean operations involved: first, the set of reachable nodes R is computed through a symbolic visit of the graph, then, starting from $R \times R$ all the pairs $\langle a, b \rangle$ for which it is possible to prove that a is not bisimilar to b are removed. In (BdS92) experimental results on the performances of the algorithm are presented, while there is no a deep discussion of its complexity in terms of basic OBDD operations.

In (FV99) Fisler and Vardi analyze the complexity of the symbolic versions of the algorithms of Paige and Tarjan (PT87); Bouajjani, Fernandez, and Halbwachs (BFH90); and Lee and Yannakakis (LY92). In particular, they determine the number of basic symbolic operations involved in each iteration by the three algorithms and they conclude, through experimental results, that an optimized version of the algorithm in (PT87) (splitting only reachable blocks) performs better than the other two algorithms, since it gains from the *right* choice of the splitters.

In (DGPP02) a symbolic version of the algorithm previously presented in (DPP01) is proposed. The main problem in making the algorithm in (DPP01) symbolic is that the strongly connected components are used to compute the ranks and, in the symbolic case, the computation of the strongly connected components would require $O(|N| \log |N|)$ operations if we admit to use only a constant num-

ber of OBDDs (BGS00)). For this reason in (DGPP02) is described a procedure which computes the ranks in $O(|N|)$ steps avoiding the computation of the strongly connected components. Notice that it is possible to compute the strongly connected components in $O(|N|)$ steps, but using in the worst case p OBDDs, where p is the length of the longest acyclic path in the graph (see (GPP03)).

In (HHK95) a symbolic version of their simulation algorithm is also discussed. Even if in (BG00) the authors do not consider this issue, their algorithm seems to be particularly suitable for a symbolic implementation, since it always works on classes of nodes. Such a nice feature is shared also by the algorithm we present in the next section.

4. A New Simulation Algorithm

In this section we study the simulation problem in a general algebraic environment. This study lead us to the definition of a generalization of the coarsest partition problem which is at the basis of our time/space efficient simulation algorithm. A preliminary version of the algorithm we describe here has been presented in (GPP02a).

4.1. Simulations as Partitioning Problems

In Section 3.2 we observed that in the case of simulation labels are fundamental: it is not possible to completely remove the labels. However, we begin here by showing that it is possible to encode the information in Σ (the labels) by adding new nodes to the labelled graph. This encoding turns out to be useful in order to compare the problem of computing the similarity quotient with the *generalized partitioning problem* we will present later in this section.

Our encoding allows us to start with a partition in which all the nodes of the original labelled graph are in the same class (i.e., they have the same label).

DEFINITION 4.1 (Unlabelling). *Let $G = (N, \to, \Sigma)$ be a labelled graph. Consider the following sets:*

$$\begin{aligned} L &= \{\ell_\alpha \mid \alpha \in \Sigma\} \\ N' &= N \cup L \\ \to' &= \to \cup \{\langle a, \ell_\alpha \rangle \mid a \in \alpha \in \Sigma\} \\ \Sigma' &= \{N\} \cup \{\{\ell_\alpha\} \mid \ell_\alpha \in L\}. \end{aligned}$$

We call the labelled graph $G' = (N', \to', \Sigma')$ the unlabelling *of G.*

In Figure 3 we give an example of a graph together with its unlabelling.

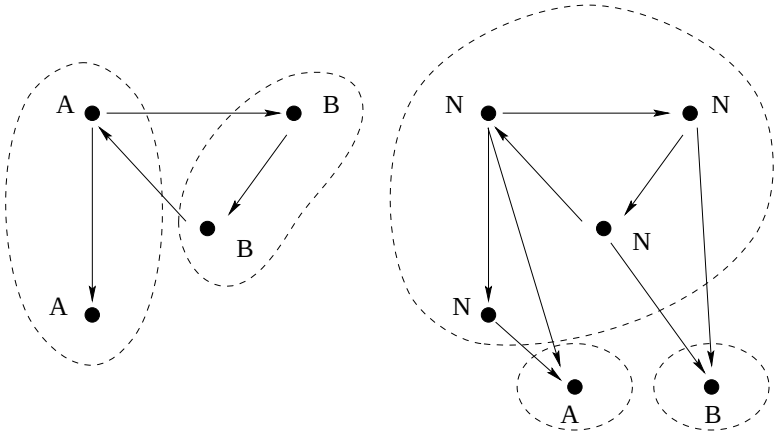

Figure 3. Unlabelling Example.

DEFINITION 4.2. *Let G be a labelled graph and G' be its unlabelling. If $\leq^{G'}$ is a simulation over G', then we define (its restriction to G)*

$$\leq^{G} = r(\leq^{G'}) = \leq^{G'} \restriction N.$$

If $\leq^{G}$ is a simulation over G, then we define (its extension to G')

$$\leq^{G'} = e(\leq^{G}) = \leq^{G} \cup \{\langle \ell_\alpha, \ell_\alpha \rangle \mid \ell_\alpha \in L\}.$$

LEMMA 4.3. *Let G be a labelled graph and let G' be its unlabelling. If $\leq^{G'}$ is a simulation over G', then its restriction to N, $\leq^{G} = r(\leq^{G'})$, is a simulation over G.*
If $\leq^{G}$ is a simulation over G, then its extension to N', $\leq^{G'} = e(\leq^{G})$ is a simulation over G'.
Moreover, it holds that:

1. *if $\leq_1^{G'} \subseteq \leq_2^{G'}$, then $r(\leq_1^{G'}) \subseteq r(\leq_2^{G'})$;*
2. *if $\leq_1^{G} \subseteq \leq_2^{G}$, then $e(\leq_1^{G}) \subseteq e(\leq_2^{G})$;*
3. *$r(e(\leq^{G})) = \leq^{G}$;*
4. *$e(r(\leq^{G'})) = \leq^{G'}$.*

The above lemma establishes a link among the simulations on a graph and the simulations on its unlabelling which is used to prove that from the maximum simulation on the unlabelling we can immediately obtain the maximum simulation on the original graph.

PROPOSITION 4.4. *Let G be a labelled graph and let G' be its unlabelling. Let $\leq_s^{G'}$ be the maximal simulation over G', then its restriction to N is the maximal simulation over G.*

Let $\leq_s^{G}$ be the maximal simulation over G, then its extension to N' is the maximal simulation over G'.

We are now ready to introduce the *Generalized Coarsest Partition Problem* (GCPP) which is the central notion in our approach. The *generalized* comes from the fact that we are not only going to deal with partitions to be refined, as in the classical coarsest partition problems (see Section 3.1), but with *pairs* in which we have a partition and a relation over the partition. The equivalence between the simulation problem and the GCPP will be proved at the end of this section.

DEFINITION 4.5 (Partition pair). *Let $G = (N, \rightarrow)$ be a finite graph. A* partition pair *over G is a pair $\langle \Sigma, P \rangle$ in which Σ is a partition over N, and $P \subseteq \Sigma^2$ is a reflexive and acyclic relation over Σ.*

Notice that a labelled graph $G = (N, \rightarrow, \Sigma)$ can be seen as a graph $G' = (N, \rightarrow)$ together with the partition pair $\langle \Sigma, I \rangle$, where I is the identity relation over Σ.

Given two partitions Π and Σ, such that Π is finer than Σ (i.e., each block of Π is included in a block of Σ), and a relation P over Σ, we use the notation $P(\Pi)$ to refer to the relation *induced* on Π by P, i.e.:

$$\forall \alpha\beta \in \Pi((\alpha, \beta) \in P(\Pi) \Leftrightarrow \exists \alpha'\beta'((\alpha', \beta') \in P \wedge \alpha \subseteq \alpha' \wedge \beta \subseteq \beta')).$$

Denoting by $\mathcal{P}(G)$ the set of partition pairs over G we now introduce the partial order over which we will define the notion of coarsest partition pair.

DEFINITION 4.6. *Let $\langle \Sigma, P \rangle, \langle \Pi, Q \rangle \in \mathcal{P}(G)$:*

$$\langle \Pi, Q \rangle \sqsubseteq \langle \Sigma, P \rangle \quad \textit{iff} \quad \Pi \textit{ is finer than } \Sigma \textit{ and } Q \subseteq P(\Pi).$$

The search for the coarsest partition pair will proceed on the following structures (graphs):

DEFINITION 4.7 ($\rightarrow_\exists, \rightarrow_\forall$ and Quotient Structures). *Let $G = (N, \rightarrow)$ be a graph, and Π a partition of N. The* $\exists$-quotient structure *over Π is the graph $\Pi_\exists = (\Pi, \rightarrow_\exists)$, where*

$$\alpha \rightarrow_\exists \gamma \quad \textit{iff} \quad \exists a \exists c(a \in \alpha \wedge c \in \gamma \wedge a \rightarrow c).$$

The $\forall$-quotient structure *over Π is the graph $\Pi_\forall = (\Pi, \rightarrow_\forall)$, where*

$$\alpha \rightarrow_\forall \gamma \quad \textit{iff} \quad \forall a(a \in \alpha \Rightarrow \exists c(c \in \gamma \wedge a \rightarrow c)).$$

The $\exists\forall$-quotient structure *over Π is the structure $\Pi_{\exists\forall} = (\Pi, \rightarrow_\exists, \rightarrow_\forall)$.*

Notice that it is always $\rightarrow_\forall \subseteq \rightarrow_\exists$. In other worlds $\alpha \rightarrow_\forall \gamma$ is a shorthand for $\alpha \subseteq \rightarrow^{-1}(\gamma)$, while $\alpha \rightarrow_\exists \gamma$ stands for $\alpha \cap \rightarrow^{-1}(\gamma) \neq \emptyset$. Similar notations (called *relation transformers* and *abstract transition relations*) were introduced in (DGG97) in order to combine Model Checking and Abstract Interpretation.

We introduce now a definition strongly connected to the definition of the quotient structures, to be used later in our algorithm (see Section 4.3).

DEFINITION 4.8 (Induced Structure). *Given a graph, $G = (N, \rightarrow)$, and two partitions of N, Σ and Π with Π finer then Σ the* $\exists\forall$-induced quotient structure *over Π is the structure*

$$\Sigma_{\exists\forall}(\Pi) = (\Pi, \rightarrow_\exists^\Sigma(\Pi), \rightarrow_\forall^\Sigma(\Pi))),$$

where:

$$\begin{array}{ll} \alpha \rightarrow_\exists^\Sigma (\Pi)\gamma & \textit{iff} \quad \alpha \cap \rightarrow^{-1}(\gamma) \neq \emptyset \\ \alpha \rightarrow_\forall^\Sigma (\Pi)\gamma & \textit{iff} \quad \alpha \rightarrow_\exists^\Sigma (\Pi)\gamma \wedge \alpha \subseteq \rightarrow^{-1}(\gamma') \wedge \gamma \subseteq \gamma' \in \Sigma \end{array}$$

with $\alpha, \beta \in \Pi$.

We are now ready to introduce the fundamental notion in the generalized coarsest partition problems, the notion of stability of a partition pair with respect to a relation.

DEFINITION 4.9 (Stability). *Given a graph $G = (N, \rightarrow)$, we say that a partition pair $\langle \Sigma, P \rangle$ is* stable *with respect to the relation $\rightarrow$ if and only if*

$$\forall \alpha, \beta, \gamma \in \Sigma((\alpha, \beta) \in P \wedge \alpha \rightarrow_\exists \gamma \Rightarrow \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \beta \rightarrow_\forall \delta)).$$

The condition in the definition of stability is equivalent to:

$$\begin{array}{l} \forall \alpha, \beta, \gamma \in \Sigma((\alpha, \beta) \in P \wedge \alpha \cap \rightarrow^{-1}(\gamma) \neq \emptyset \Rightarrow \\ \qquad\qquad \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \beta \subseteq \rightarrow^{-1}(\delta))). \end{array}$$

As it will be proved (see Theorem 4.17) the stability condition is exactly the condition holding between two classes of $N/\equiv_s$: if $\alpha, \beta \in N/\equiv_s$ with $\alpha \leq_s \beta$ (i.e. all the elements of α are simulated by all the elements of β) and an element a in α reaches an element c in γ, then all the elements b of β must reach at least one element d which simulates c. In particular, considering all the $\leq_s$-maximal elements d simulating c reached by elements in β, we have that all the elements in β reach a class δ which simulates c and, hence, which simulates γ.

In Figure 4 we give a graphical representation of the notion of stability in both the input and the induced structure.

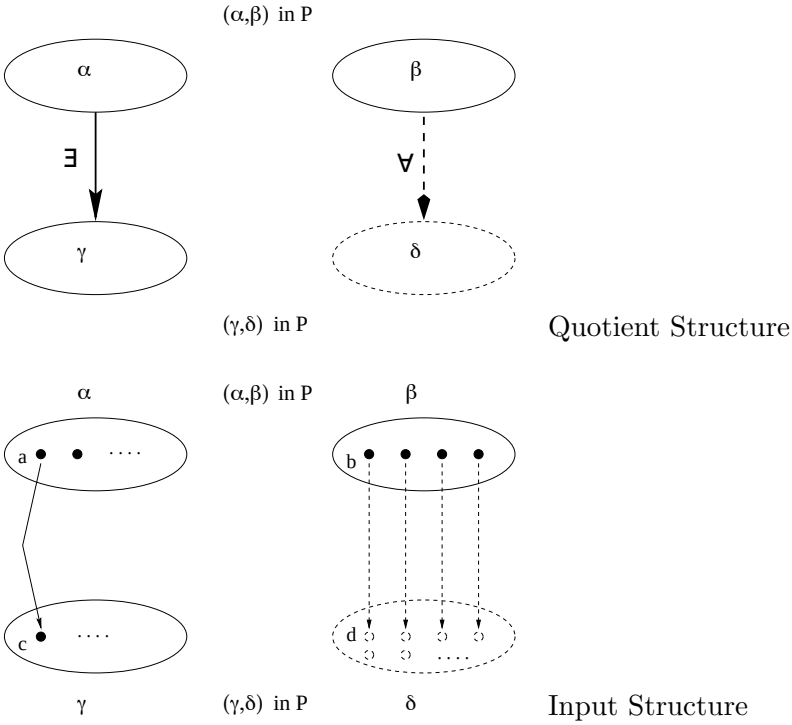

Figure 4. The Stability Condition.

We use the notion of stability of a partition pair with respect to a relation in the definition of generalized coarsest partition problem, in the same way as the notion of stability of a partition with respect to a relation is used in the definition of coarsest partition problem.

DEFINITION 4.10 (Generalized Coarsest Partition Problem). *Given a graph $G = (N, \rightarrow)$ and partition pair $\langle \Sigma, P \rangle$ over G the* generalized coarsest partition problem *consists in finding a partition pair $\langle S, \preceq \rangle$ such that:*

(a) $\langle S, \preceq \rangle \sqsubseteq \langle \Sigma, P^+ \rangle$;

(b) $\langle S, \preceq \rangle$ is stable with respect to $\rightarrow$;

(c) $\langle S, \preceq \rangle$ is $\sqsubseteq$-maximal satisfying (a) and (b).

If $\langle S, \preceq \rangle$ is a partition pair which satisfies (a) and (b) we say that it is a stable refinement *of $\langle \Sigma, P \rangle$.*

Notice that in the above definition $\langle S, \preceq \rangle$ is a refinement of $\langle \Sigma, P^+ \rangle$, while it can be the case that it is not a refinement of $\langle \Sigma, P \rangle$. In particular this happens always in case $\langle \Sigma, P \rangle$ is not stable while $\langle \Sigma, P^+ \rangle$ is

stable. In this case $\langle \Sigma, P^+ \rangle$ itself is a solution of the GCPP. In general, the solution of the GCPP can always be found by a suitable sequence of splits (of classes) and adequate completion of the relation in order to take the newly generated classes into account.

REMARK 4.11. *Notice that it is important that $\preceq$ is reflexive (which is the case since $\langle S, \preceq \rangle$ is a partition pair). This is necessary in order to prove that there is always a unique maximal solution. Consider the graph $G = (N, \rightarrow)$ with $N = \{a, b\}$ and $\rightarrow = \{(a, b)\}$, the partition pair $\langle \Sigma, P \rangle$ with $\Sigma = \{N\}$ and $P = \{(N, N)\}$. It holds that both $\langle \Sigma, \emptyset \rangle$ and $\langle \Pi, Q \rangle$ with $\Pi = \{\{a\}, \{b\}\}$ and $Q = \{(\{a\}, \{a\}), (\{b\}, \{b\}), (\{b\}, \{a\})\}$ are maximal solution of the partition problem, but the first is not a partition pair. Similarly the acyclicity condition is important because otherwise we could coarsen the partition merging all classes which form cycles.*

Our next task is to prove that a given GCPP has always a unique solution. To this end we will spell out the connection between the similarity quotient problems and the GCPP's introducing generalized unlabellings. The generalized unlabelling of $G = (N, \rightarrow)$ and $\langle \Sigma, I \rangle$, where I is the identity relation over Σ, will turn out to be exactly the unlabelling of $(N, \rightarrow, \Sigma\rangle$ (see Definition 4.3). In this sense the *generalized unlabelling* is a generalization of the unlabelling for the case in which we have to deal with a partition pair and not only with a partition.

DEFINITION 4.12 (Generalized unlabelling). *Let $G = (N, \rightarrow)$ be a graph and $\langle \Sigma, P \rangle$ be a partition pair over G. Consider the labelled graph $G' = (N', \rightarrow', \Sigma')$ defined as:*

$$\begin{aligned} L &= \{\ell_\alpha \mid \alpha \in \Sigma\} \\ N' &= N \uplus L \\ \rightarrow' &= \rightarrow \cup \{\langle a, \ell_\alpha \rangle \mid a \in \beta \in \Sigma \wedge (\alpha, \beta) \in P^+\} \\ \Sigma' &= \{N\} \cup \{\{\ell_\alpha\} \mid \ell_\alpha \in L\}. \end{aligned}$$

We call the labelled graph G' the generalized unlabelling *of G and $\langle \Sigma, P \rangle$.*

Here we have to use P^+ in the definition of $\rightarrow'$ in order to produce a simulation from a stable refinement, since α-nodes can simulate β-nodes whenever $(\alpha, \beta) \in P^+$. In Figure 5 we present a partition pair together with its generalized unlabelling.

Now we prove the strong connection between stable refinements of $\langle \Sigma, P \rangle$ and simulations over G'.

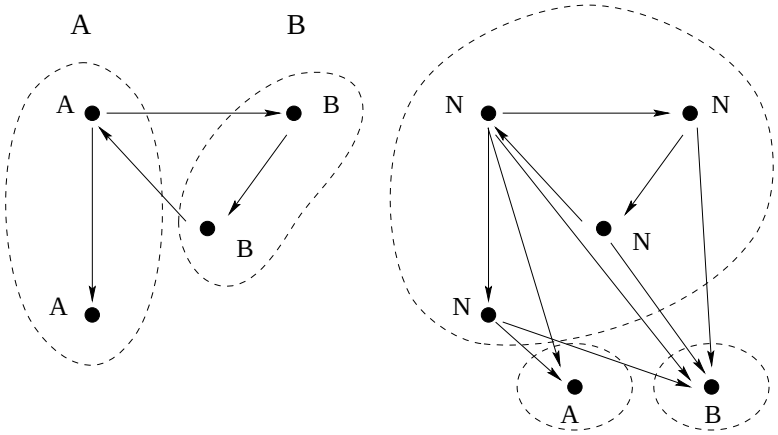

Figure 5. Generalized Unlabelling Example.

DEFINITION 4.13. *Let $G = (N, \rightarrow)$ be a graph, $\langle \Sigma, P \rangle$ a partition pair over G, and G' the generalized unlabelling of G and $\langle \Sigma, P \rangle$. Consider the set S_{GP} of the stable (with respect to $\rightarrow$) refinements of $\langle \Sigma, P \rangle$ and the set S_{sim} of the simulations over G'. We define the two functions $f_1 : S_{GP} \rightarrow S_{sim}$ and $f_2 : S_{sim} \rightarrow S_{GP}$ as follows*[2]*:*

$$f_1(\langle \Pi, Q \rangle) = \leq_{\langle \Pi, Q \rangle}$$
$$\forall a, b \in N(a \leq_{\langle \Pi, Q \rangle} b \text{ iff } a \in \alpha \in \Pi \wedge b \in \beta \in \Pi \wedge (\alpha, \beta) \in Q)$$
$$\forall \ell_\alpha \in L(\ell_\alpha \leq_{\langle \Pi, Q \rangle} \ell_\alpha)$$

$$f_2(\leq) = \langle \Pi_s, Q_s \rangle$$
$$\forall a \in N([a] = \{b \mid a \leq^* b \leq^* a\} \in \Pi_s)$$
$$(\alpha, \beta) \in Q_s \text{ iff } \exists a \in \alpha \exists b \in \beta (a \leq^* b)$$

Notice that the condition $\exists a \in \alpha \exists b \in \beta(a \leq^* b)$ is equivalent to $\forall a \in \alpha \forall b \in \beta(a \leq^* b)$, since we are using the (reflexive and) transitive closure of the simulation $\leq$. Notice also that it is necessary to prove that the codomains of f_1 and f_2 are correctly defined. We prove this fact and some properties of f_1 and f_2 in the following Lemma.

LEMMA 4.14. *The function f_1 and f_2 are such that:*

1. *if $\langle \Pi, Q \rangle \in S_{GP}$, then $f_1(\langle \Pi, Q \rangle) \in S_{sim}$;*
2. *if $\leq \in S_{sim}$, then $f_2(\leq) \in S_{GP}$;*
3. *if $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$, then $f_1(\langle \Pi_1, Q_1 \rangle) \subseteq f_1(\langle \Pi_2, Q_2 \rangle)$;*

[2] $\leq^*$ is the reflexive and transitive closure of $\leq$.

4. if $\leq^1 \subseteq \leq^2$*, then* $f_2(\leq^1)) \sqsubseteq f_2(\leq^2)$*;*

5. $\leq \subseteq f_1(f_2(\leq))$*;*

6. $\langle \Pi, Q\rangle \sqsubseteq f_2(f_1(\langle \Pi, Q\rangle))$*.*

THEOREM 4.15 (Existence and Uniqueness). *Given* $G = (N, \rightarrow)$ *and a partition pair* $\langle \Sigma, P\rangle$ *over* G*, the GCPP over* G *and* $\langle \Sigma, P\rangle$ *has always a unique solution.*

COROLLARY 4.16. *If* $\langle S, \preceq\rangle$ *is the solution of the GCPP over* G *and* $\langle \Sigma, P\rangle$*, then* $\preceq$ *is a partial order over* S*.*

Proof. We have to prove that $\preceq$ is transitive. This is an immediate consequence of the fact that $\langle S, \preceq\rangle = f_2(\leq_s)$, where $\leq_s$ is the maximal simulation over G' (generalized unlabelling). □

We proved that each generalized coarsest partition problem has a unique solution which can be determined by solving a similarity quotient problem. The following easily proved result states that the converse also hold.

THEOREM 4.17 (Simulation as GCPP). *Let* $G = (N, \rightarrow, \Sigma)$ *be a labelled graph. Let* $\langle S, \preceq\rangle$ *be the solution of the GCPP over* $G^- = (N, \rightarrow)$ *and* $\langle \Sigma, I\rangle$*, where* I *is the identity relation over* Σ*. Then,* S *is the simulation quotient of* G*, i.e.* $S = N/\equiv_s$*, and* $f_1(\langle S, \preceq\rangle)$ *is the maximal simulation over* G*.*

Proof. As we already noticed, since the relation over Σ is the identity relation, we have that the generalized unlabelling G' of G^- is exactly the unlabelling of G. Hence from Lemma 4.14 and Proposition 4.4 we immediately obtain the thesis. □

The results in this section guarantee that in order to solve the problem of determining the simulation quotient of a labelled graph $G = (N, \rightarrow, \Sigma)$ we can equivalently solve the generalized coarsest partition problem over $(N, \rightarrow)$ and $\langle \Sigma, I\rangle$. If $\langle S, \preceq\rangle$ is the solution of the GCPP, then the relation $\leq_{\langle S, \preceq\rangle}$ defined as

$$\forall a, b \in N(a \leq_{\langle S, \preceq\rangle} b \text{ iff } [a] \preceq [b])$$

is the maximal simulation over G, and S is the partition which corresponds to the sim-equivalence $\equiv_s$.

4.2. Solving the Generalized Coarsest Partition Problems

To give an operational content to the results in the previous section, we now introduce an operator σ, mapping partition-pairs into partition-pairs, which will turn out to be the engine of our algorithm. A procedure which compute σ will be used to solve GCPP's and, hence, to compute similarity quotients: it is only necessary to iterate the computation of σ at most $|S|$ times.

In particular, the operator σ is defined in such a way that it refines the partition-pair $\langle \Sigma, P\rangle$ obtaining a partition-pair $\langle \Pi, Q\rangle$ which is *more stable* than $\langle \Sigma, P\rangle$ and is never finer than the solution of the GCPP over $\langle \Sigma, P\rangle$.

In the first condition of the definition of σ we impose to split the classes of Σ which do not respect the stability condition with respect to themselves: if a class α is such that $\alpha \rightarrow_\exists \gamma$ and it does not exists a class δ such that $(\gamma, \delta) \in P$ and $\alpha \rightarrow_\forall \delta$, then the pair (α, α) does not respect the stability condition, hence we must split α. The first condition is used to build Π, while the second and the third conditions in σ are used to define Q using the Π already obtained. Intuitively, the second and the third conditions allow to obtain Q from P by starting from $P(\Pi)$ and removing the minimum number of pairs which contradict stability.

DEFINITION 4.18 (Operator σ). *Let $G = (N, \rightarrow)$ be a graph and $\langle \Sigma, P\rangle$ be a partition pair over G, the partition pair $\langle \Pi, Q\rangle = \sigma(\langle \Sigma, P\rangle)$ is defined as:*

(1σ) *Π is the coarsest partition finer than Σ such that*

$$(a)\ \ \forall \alpha \in \Pi\, \forall \gamma \in \Sigma(\alpha \rightarrow_\exists \gamma \Rightarrow \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \alpha \rightarrow_\forall \delta));$$

(2σ) *Q is maximal such that $Q \subseteq P(\Pi)$ and if $(\alpha, \beta) \in Q$, then:*

$$(b)\ \ \forall \gamma \in \Sigma(\alpha \rightarrow_\forall \gamma \Rightarrow \exists \gamma' \in \Sigma((\gamma, \gamma') \in P \wedge \beta \rightarrow_\exists \gamma')) \quad \textit{and}$$
$$(c)\ \ \forall \gamma \in \Pi(\alpha \rightarrow_\forall \gamma \Rightarrow \exists \gamma' \in \Pi((\gamma, \gamma') \in Q \wedge \beta \rightarrow_\exists \gamma')).$$

By abuse of notation we use $\rightarrow_\exists$ and $\rightarrow_\forall$ also when the classes belong to different partitions.

Condition (a) in (1σ) imposes to suitably split the classes of the partition Σ: these splits are forced by the fact that we are looking for a partition-pair *stable* with respect to the relation of the given graph and we know that in each partition-pair $\langle \Sigma, P\rangle$ the second component is reflexive (i.e. $\forall \alpha \in \Sigma(\alpha, \alpha) \in P$). Using condition (b) in (2σ), together with condition (a) in (1σ) and exploiting the fact that P is acyclic, it is possible to prove that Q is acyclic: the acyclicity of P ensures that a cycle could arise only among classes of Π which are all contained in a unique class of Σ, then using condition (a) in (1σ) and (b) in (2σ)

we obtain a contradiction. Condition (c) is fundamental, together with condition (a), in order to obtain the result in Theorem 4.24: if it holds that $\alpha \rightarrow_\forall \gamma$, then *no matter how we split* α one of the subclasses generated from α has a chance (in the solution of GCPP) to be in relation with at least one of the subclasses generated from β only if $\beta \rightarrow_\exists \gamma'$ and γ is in relation with γ'. The following results guarantee the correctness of σ and, hence, of our approach.

LEMMA 4.19 (Existence). *Let $G = (N, \rightarrow)$ be a graph and $\langle \Sigma, P \rangle$ be a partition pair over G. There exists at least a partition pair $\langle \Pi, Q \rangle$ which satisfies the conditions in Definition 4.18, i.e. σ is always defined.*

THEOREM 4.20 (Uniqueness). *Let $G = (N, \rightarrow)$ be a graph and $\langle \Sigma, P \rangle$ be a partition pair over G. There exists a unique maximal pair $\langle \Pi, Q \rangle$ which satisfies the conditions in Definition 4.18, i.e. σ is always uniquely defined.*

Fix points of the σ operator can be used to compute the solution of the GCPP.

LEMMA 4.21. *Let $\langle S, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. If $\langle S, \preceq \rangle \sqsubseteq \langle \Pi, Q \rangle$, then $\langle S, \preceq \rangle \sqsubseteq \sigma(\langle \Pi, Q \rangle)$.*

LEMMA 4.22. *Let $G = (N, \rightarrow)$ be a graph and $\langle \Pi, Q \rangle$ be a partition pair over G. Let $\langle S, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Pi, Q \rangle$. If $\sigma(\langle \Pi, Q \rangle) = \langle \Pi, Q \rangle$, then $\langle \Pi, Q^+ \rangle = \langle S, \preceq \rangle$.*

The following lemma is an immediate consequence of the definition of GCPP.

LEMMA 4.23. *Let $G = (N, \rightarrow)$ be a graph and $\langle \Pi, Q \rangle \sqsubseteq \langle \Sigma, P \rangle$ be two partition pairs. Let $\langle S, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. If $\langle S, \preceq \rangle \sqsubseteq \langle \Pi, Q^+ \rangle$, then $\langle S, \preceq \rangle$ is also the solution of the GCPP over G and $\langle \Pi, Q \rangle$.*

The following theorem, whose proof is now fairly standard, concludes our chain of reasonings.

THEOREM 4.24 (Fix Point). *Let $G = (N, \rightarrow)$ be a graph and $\langle \Sigma, P \rangle$ a partition pair over G with P transitive. Let $\langle S, \preceq \rangle$ be the solution of the GCPP over G and $\langle \Sigma, P \rangle$. If n is such that $\sigma^n(\langle \Sigma, P \rangle) = \sigma^{n+1}(\langle \Sigma, P \rangle)$, then $\sigma^n(\langle \Sigma, P \rangle) = \langle S, \preceq \rangle$.*

The meaning of this theorem is that if $\langle \Sigma, P \rangle$ is a partition pair over a graph G such that P is transitive, then there exists i such that

$\sigma^i(\langle\Sigma, P\rangle) = \sigma^{i+1}(\langle\Sigma, P\rangle)$ and $\sigma^i(\langle\Sigma, P\rangle)$ is the solution of the GCPP over G and $\langle\Sigma, P\rangle$.

Estimating how large is the index i in the worst case allows us to point out another important property of the operator σ. When we iteratively apply the operator σ until we reach a fix point, at each iteration we refine a partition and we remove pairs from a relation. What we prove in the following theorem is that is never the case that there exists an iteration in which we do not refine the partition, but we remove pairs from the relation. In a certain sense this means that the two conditions we have given in (2σ) to remove pairs are *optimal*.

THEOREM 4.25 (Complexity). *Let $G = (N, \rightarrow)$ be a graph and $\langle\Sigma, P\rangle$ a partition pair over G with P transitive. Let $\langle S, \preceq\rangle$ be the solution of the GCPP over G and $\langle\Sigma, P\rangle$. If i is such that $\sigma^i(\langle\Sigma, P\rangle) = \langle\Sigma_i, \mathsf{P}_i\rangle$ and $\sigma^{i+1}(\langle\Sigma, P\rangle) = \langle\Sigma_i, \mathsf{P}_{i+1}\rangle$, then $\mathsf{P}_{i+1} = \mathsf{P}_i$ and $\langle\Sigma_i, \mathsf{P}_i\rangle = \langle S, \preceq\rangle$.*

Hence from this Theorem, since σ never adds pairs to the relations, we can conclude that if P is transitive and $\Sigma_{i+1} = \Sigma_i$, then $\mathsf{P}_{i+1} = \mathsf{P}_i$, this last is transitive, and $\langle\Sigma_i, \mathsf{P}_i\rangle = \sigma^i(\langle\Sigma, P\rangle)$ is the solution of the GCPP on G and $\langle\Sigma, P\rangle$. In the case that P is not transitive we obtain the same result starting from P^+ instead of P. This gives us an upper bound to the index i which is at most $|\Sigma_i|$, this last is in the worst case $O(|N|)$.

COROLLARY 4.26. *The solution $\langle S, \preceq\rangle$ of the GCPP over a graph G and a partition pair $\langle\Sigma, P\rangle$ can be computed computing at most $|S|$ times the operator σ.*

Proof. From Theorem 4.25 we obtain that

$$\langle S, \preceq\rangle = \sigma^{|S|}(\langle\Sigma, P\rangle).$$

□

The characterizations obtained in this section allow us to conclude that if we are able to define a procedure which given a partition-pair $\langle\Sigma, P\rangle$ computes $\sigma(\langle\Sigma, P\rangle)$, then we can use it to solve GCPP's and hence to compute similarity quotients. In particular, we recall that given a similarity quotient problem over a labelled graph $G = (N, \rightarrow, \Sigma)$ in order to solve it it is sufficient solve the GCPP over G and $\langle\Sigma, I\rangle$ (notice that I is trivially transitive).

Moreover, the last corollary ensures that it will be necessary to iterate the procedure which computes σ at most $|S|$ times. This is a first improvement on time complexity with respect to the algorithm presented in (BG00): in a certain sense (BG00) computes an operator

Partitioning Algorithm$((N, \rightarrow), \langle \Sigma, P \rangle)$

```
change := ⊤;
i := 0;
while change do
        change := ⊥;
        Σ_{i+1} := Refine(Σ_i, P_i, change);
        P_{i+1} := Update(Σ_i, P_i, Σ_{i+1});
        i := i + 1;
```

Figure 6. The **Partitioning Algorithm**.

which refines the partition-pair less than σ, and hence it is possible that the computation has to be iterated up to $|S|^2$ times.

In the next section we present the procedure which computes σ.

4.3. Partitioning Algorithm

In this section we propose an algorithm which solves the generalized coarsest partition problem we have presented in the previous section. The **Partitioning Algorithm** takes as input a graph $G = (N, \rightarrow)$ and a partition pair $\langle \Sigma, P \rangle$, with P transitive, calls the two functions **Refine** and **Update** until a fix point is reached, and returns the partition pair $\langle S, \preceq \rangle$ which is the solution of the GCPP over G and $\langle \Sigma, P \rangle$. In order to solve the GCPP over G and $\langle \Sigma, P \rangle$, with P not transitive, it is sufficient to first compute P^+, the cost of this operation is $O(\Sigma^3)$, and hence it does not affect the global cost of the algorithm. The function **Refine** takes as input a partition pair $\langle \Sigma_i, P_i \rangle$ and it returns the partition Σ_{i+1} which is the coarsest that satisfies the condition in (1σ) of Definition 4.18. The function **Update** takes as input a partition pair $\langle \Sigma_i, P_i \rangle$ and a the refinement Σ_{i+1} and it produces the acyclic and reflexive relation over Σ_{i+1} which is the greatest satisfying the conditions in (2σ) of Definition 4.18. In particular, at the end of each iteration of the while-loop in the **Partitioning Algorithm** we have that $\langle \Sigma_{i+1}, P_{i+1} \rangle = \sigma(\langle \Sigma_i, P_i \rangle)$. It is immediate to see that the **Refine** function works exactly as described in the proof of Theorem 4.20, and hence it produce the partition which is the first element of $\sigma(\Sigma_i, \mathsf{P}_i)$.

COROLLARY 4.27. *If* $\sigma(\langle \Sigma_i, \mathsf{P}_i \rangle) = \langle \Pi, Q \rangle$, *then* $\Pi = \Sigma_{i+1}$.

Update removes pairs from $(\mathsf{P}_i)(\Sigma_{i+1}) = \mathsf{Ind}_{i+1}$ in order to obtain the relation P_{i+1} which satisfies the two conditions in (2σ) of Definition 4.18. In particular Ref_{i+1} satisfies the first of the two conditions, but not necessarily the second one.

Refine$(\Sigma_i, \mathsf{P}_i, \mathsf{change})$

$\Sigma_{i+1} := \Sigma_i$;
for each $\alpha \in \Sigma_{i+1}$ do $\mathsf{Stable}(\alpha) := \emptyset$;
for each $\gamma \in \Sigma_i$ do $\mathsf{Row}(\gamma) := \{\gamma' \mid (\gamma, \gamma') \in \mathsf{P}_i)\}$;
Let Sort be a topological sorting of $\langle \Sigma_i, \mathsf{P}_i \rangle$;
while $\mathsf{Sort} \neq \emptyset$ do
    $\gamma := \mathit{dequeue}(\mathsf{Sort})$;
    $A := \emptyset$;
    for each $\alpha \in \Sigma_{i+1}, \alpha \rightarrow_\exists \gamma, \mathsf{Stable}(\alpha) \cap \mathsf{Row}(\gamma) = \emptyset$ do
        $\alpha_1 := \alpha \cap \rightarrow^{-1}(\gamma)$;
        $\alpha_2 := \alpha \setminus \alpha_1$;
        if $\alpha_2 \neq \emptyset$ then $\mathsf{change} := \top$;
        $\Sigma_{i+1} := \Sigma_{i+1} \setminus \{\alpha\}$;
        $A := A \cup \{\alpha_1, \alpha_2\}$;
        $\mathsf{Stable}(\alpha_1) := \mathsf{Stable}(\alpha) \cup \{\gamma\}$;
        $\mathsf{Stable}(\alpha_2) := \mathsf{Stable}(\alpha)$;
    $\Sigma_{i+1} := \Sigma_{i+1} \cup A$;
    $\mathsf{Sort} := \mathsf{Sort} \setminus \{\gamma\}$;
return Σ_{i+1}

Figure 7. The **Refine** Function.

Update$(\Sigma_i, \mathsf{P}_i, \Sigma_{i+1})$

$\mathsf{Ind}_{i+1} := \{(\alpha_1, \beta_1) \mid \alpha_1, \beta_1 \in \Sigma_{i+1}, \alpha_1 \subseteq \alpha, \beta_1 \subseteq \beta (\alpha, \beta) \in \mathsf{P}_i\}$;
$\Sigma_{i\exists\forall}(\Sigma_{i+1}) := \langle \Sigma_{i+1}, \rightarrow_\exists^{\Sigma_i}(\Sigma_{i+1}), \rightarrow_\exists^{\Sigma_i}(\Sigma_{i+1}) \rangle$;
$\mathsf{Ref}_{i+1} := \mathbf{New\_HHK}(\Sigma_{i\exists\forall}(\Sigma_{i+1}), \mathsf{Ind}_{i+1}, \bot)$;
$\Sigma_{(i+1)\exists\forall} := \langle \Sigma_{i+1}, \rightarrow_\exists, \rightarrow_\forall \rangle$;
$\mathsf{P}_{i+1} := \mathbf{New\_HHK}(\Sigma_{(i+1)\exists\forall}, \mathsf{Ref}_{i+1}, \top)$;
return P_{i+1}

Figure 8. The **Update** Function.

The deletion of "wrong" pairs is performed calling the function **New_HHK**, which is a version of (HHK95) adapted to our purposes here.

Notice that the space complexity of the calls to the adapted version of (HHK95) remains limited since they are made on quotient stuctures. This function is based on the use of the two structures $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ (cf. Definition 4.8) and $\Sigma_{i+1\exists\forall}$ (cf. Definition 4.7), and on the following equivalent formulation of the second condition in (2σ).

PROPOSITION 4.28. *Let $G = (N, \rightarrow)$ be a graph, $\langle \Sigma, P \rangle$ be a partition pair and Π be a partition finer than Σ. Q satisfies (2σ) of Definition 4.18 if and only if Q is the maximal relation over Π such that $Q \subseteq P(\Pi)$*

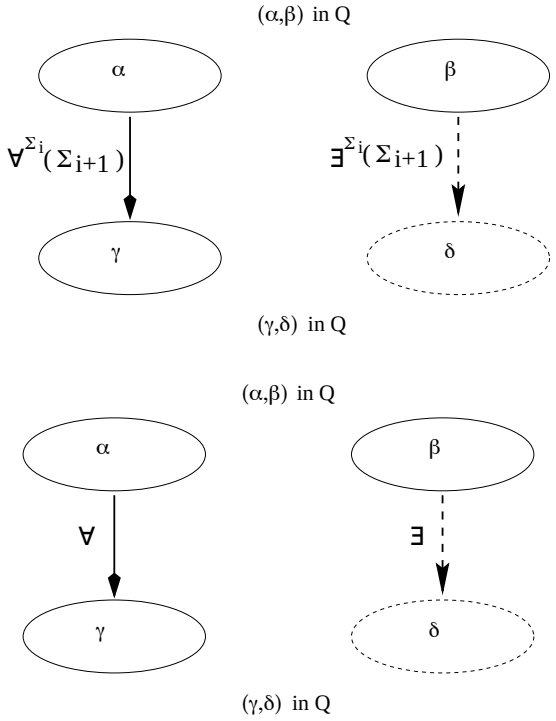

Figure 9. The conditions in (2σ) on the quotient structures.

and if $(\alpha, \beta) \in Q$*, then:*

$$\forall\gamma \in \Pi(\alpha \rightarrow^{\Sigma}_{\forall} (\Pi)\gamma \Rightarrow \exists\gamma' \in \Pi((\gamma,\gamma') \in P(\Pi) \wedge \beta \rightarrow^{\Sigma}_{\exists} (\Pi)\gamma'))$$
$$\forall\gamma \in \Pi(\alpha \rightarrow_{\forall} \gamma \Rightarrow \exists\gamma' \in \Pi((\gamma,\gamma') \in Q \wedge \beta \rightarrow_{\exists} \gamma')),$$

where $\rightarrow^{\Sigma}_{\forall} (\Pi)$ *and* $\rightarrow^{\Sigma}_{\forall} (\Pi)$ *are the edges of the* $\exists\forall$*-induced quotient structure, while* $\rightarrow_{\exists}$ *and* $\rightarrow_{\forall}$ *are the edges of the* $\exists\forall$ *quotient structure.*

In Figure 9 we present the conditions described in Proposition 4.28 on the $\exists\forall$-induced quotient structure and on the $\exists\forall$-quotient structure. As a consequence of Theorem 4.24 these conditions are weaker than the stability condition.

The computation performed by **Update** corresponds to determine the largest relation included in P_i and satisfying conditions (2σ), thereby getting us closer to stability. Such a computation is proved correct as a fix-point computation of an operator τ defined as follows:

DEFINITION 4.29. *Let* $D = (T, R_1, R_2)$ *be such that* $R_2 \subseteq R_1 \subseteq T \times T$*, and* $K \subseteq T \times T$ *a relation over* T*. We define* $\tau_D(K)$ *as:*

$$\tau_D(K) = K \setminus \{(a,b) \mid \exists c(aR_2c \wedge \forall d(bR_1d \Rightarrow (c,d) \notin K))\}.$$

```
New_HHK((T, R1, R2), K, U)
    P := K;
    for each a ∈ T do
        sim(c) := {e | (c, e) ∈ K};
        rem(c) := T \ pre1(sim(c));
    while {c | rem(c) ≠ ∅} ≠ ∅ do
        let c ∈ {c | rem(c) ≠ ∅}
        for each a ∈ pre2(c), b ∈ rem(c), b ∈ sim(a) do
            sim(a) := sim(a) \ {b};
            P := P \ {(a, b)};
            if U then for each b1 ∈ pre1(b) do
                if post1(b1) ∩ sim(a) = ∅ then
                    rem(a) := rem(a) ∪ {b1};
        rem(c) := ∅;
    return P
```

Figure 10. The **New_HHK** Function.

We use $Fix(\tau_D)(K)$ to denote the greatest fix point of τ_D smaller than K.

From the definition of τ and Proposition 4.28 we immediately get the following result:

COROLLARY 4.30. *Let $G = (N, \rightarrow)$ be a graph, and $\langle \Sigma, P \rangle$ be a partition pair. If $\sigma(\langle \Sigma, P \rangle) = \langle \Pi, Q \rangle$, then:*

$$Q = Fix(\tau_{\Pi\exists\forall})(\tau_{\Sigma_{\exists\forall}(\Pi)}(P(\Pi))).$$

We now establish a connection between the operator τ and the function **New_HHK** presented in Figure 10 .

LEMMA 4.31. *The following holds:*

$$\begin{aligned} \mathbf{New\_HHK}(D, K, \bot) &= \tau_D(K) \\ \mathbf{New\_HHK}(D, K, \top) &= Fix(\tau_D)(K). \end{aligned}$$

We are now ready to prove the invariants relative to the functions **Update** and **Refine**.

THEOREM 4.32 (Refine-Update invariants). *The following holds:*

$$\langle \Sigma_{i+1}, \mathsf{P}_{i+1} \rangle = \sigma(\langle \Sigma_i, \mathsf{P}_i \rangle).$$

Proof. This is an immediate consequence of Corollary 4.27, of Corollary 4.30, and of Lemma 4.31. □

As a consequence of Theorem 4.25 and of Corollary 4.26, since the **Partitioning Algorithm** terminates whenever $\Sigma_{i+1} = \Sigma_i$, we can conclude that the **Partitioning Algorithm** computes the solution $\langle S, \preceq \rangle$ of the GCPP over G and $\langle \Sigma, P \rangle$, with P transitive, performing at most $|S|$ iterations of the while-loop.

THEOREM 4.33 (Time complexity). *Given a graph $G = (N, \rightarrow)$ and a partition pair $\langle \Sigma, P \rangle$, with P transitive, the algorithm* **Partitioning Algorithm** *computes the solution $\langle S, \preceq \rangle$ of the GCPP over them in time $O(|S|^2| \rightarrow |)$.*

Proof. From Corollary 4.26 we have that at most $|S|$ iterations of the while-loop are performed.We are going to prove that, in each iteration of the **Partitioning Algorithm**, the **Refine** function takes $O(|\Sigma_i|| \rightarrow |) = O(|S|| \rightarrow |)$ time. The cost of the refining steps overall the **Partitioning Algorithm** is then $O(|S|^2| \rightarrow |)$.

The initialization phase (the instructions before the while-loop) in **Refine** takes time $O(|\Sigma_i|^2)$. During each iteration of the while-loop an element γ is taken out of Sort and it is never reinserted in it: as there are $|\Sigma_i|$ classes in Sort, the while-loop is iterated at most $|\Sigma_i|$ times. Once γ has been dequeued, the set $\mathsf{Split}(\gamma)$ can be computed in $O(| \rightarrow^{-1} (\gamma)|)$ time. Each other instruction in the while-loop, but the for-loop, costs $O(1)$; without considering the innermost for-loop, the global while-loop's cost in a refining step is then $O(\rightarrow^{-1} (\gamma_1) + \ldots + \rightarrow^{-1} (\gamma_{|\Sigma_i|})) + O(|\Sigma_i|) = O(| \rightarrow |)$.
$\mathsf{Split}(\gamma)$ contains at most $| \rightarrow^{-1} (\gamma)|$ elements and hence the number of for-loop's iterations is bounded by $| \rightarrow^{-1} (\gamma)|$. Assuming to have P_i represented as an $\Sigma_i \times \Sigma_i$ adjacency matrix, the check $\mathsf{Stable}(\alpha) \cap \mathsf{Row}(\gamma) = \emptyset$ can be implemented in $O(|\mathsf{Stable}(\alpha)|) = O(|\Sigma_i|)$. As far as the remaining operations in the for-loop is concerned, we observe that, for each class $\alpha \in \Sigma_{i+1}$ with $\rightarrow^{-1} (\gamma) \cap \alpha \neq \varnothing$, the sets $\alpha_1 = \rightarrow^{-1} (\gamma) \cap \alpha$ and $\alpha_2 = \alpha \backslash \alpha_1$ can be provided while computing (i.e. at the cost of computing) $\mathsf{Split}(\gamma)$: strategies similar to the ones suggested in (PT87) can be used to this purpose. For-loop's instructions involving the updating of Σ_{i+1} and the setting of the Stable sets (relative to the new Σ_{i+1} classes) can be straightfull implemented in $O(1)$. Thus the global cost of the for-loop in a refining step turn out to be $O(\rightarrow^{-1} (\gamma_1)|\Sigma_i| + \ldots + \rightarrow^{-1} (\gamma_{|\Sigma_i|})|\Sigma_i|) = O(| \rightarrow ||\Sigma_i|)$. We get that the complexity of the **Refine** function is $O(|S|^2 + |S|| \rightarrow |) = O(|S|| \rightarrow |)$ [3].

[3] We assume that $|N| = O(| \rightarrow |)$ in the graph in input: note that in the context of Model Checking, Kripke structures modeling the finite systems to validate always respect the above constraint.

In the **Update** function the cost of the initialization of Ind_{i+1} is $O(\Sigma_{i+1}^2)$. As far as the initialization of the $\exists\forall$-quotient structure and the $\exists\forall$-induced quotient structure is concerned, the following procedure builds $\Sigma_{(i+1)\exists\forall} = \langle \Sigma_{i+1}, \rightarrow_\exists, \rightarrow_\forall \rangle$ and $\Sigma_{i\exists\forall}(\Sigma_{i+1}) = \langle \Sigma_{i+1}, \rightarrow_\exists^{\Sigma_i} (\Sigma_{i+1}), \rightarrow_\exists^{\Sigma_i} (\Sigma_{i+1}) \rangle$ in $O(|\Sigma_{i+1}|^2 + |\rightarrow|)$ time (each iteration of the inner-most for-loop can be easily implemented at the cost of computing $\rightarrow^{-1}(\alpha')$):

1. for each $\alpha \in \Sigma_i$;
2. compute $\rightarrow^{-1}(\alpha)$ and sign each $\beta' \in \Sigma_{i+1}, \beta' \subseteq \rightarrow^{-1}(\alpha)$;
3. for each $\alpha' \in \Sigma_{i+1}, \alpha' \subseteq \alpha$;
4. $\rightarrow_\exists := \{\langle \alpha'\beta' \rangle \mid \beta' \in \Sigma_{i+1}, \beta' \cap \rightarrow^{-1}(\alpha') \neq \varnothing\}$;
5. $\rightarrow_\exists^{\Sigma_i} (\Sigma_{i+1}) := \{\langle \alpha'\beta' \rangle \mid \beta' \in \Sigma_{i+1}, \beta' \cap \rightarrow^{-1}(\alpha') \neq \varnothing\}$;
6. $\rightarrow_\forall := \{\langle \alpha'\beta' \rangle \mid \beta' \in \Sigma_{i+1}, \beta' \subseteq \rightarrow^{-1}(\alpha')\}$;
7. $\rightarrow_\exists := \{\langle \alpha'\beta' \rangle \mid \beta' \in \Sigma_i + 1$ has been signed in step 2, $\beta' \cap \rightarrow^{-1}(\alpha') \neq \varnothing\}$.

Without considering the two calls to the **New_HHK** function, the complexity of the updating steps overall the **Partitioning Algorithm** is then $O(|S|(|\rightarrow| + |S|^2)) = O(|S|^2|\rightarrow|)$.

In order to evaluate the cost of performing the entire set of calls to the function **New_HHK**, we can either simply apply the results in (HHK95) to a single procedure's call or trying using a sort of global argument. Following the first way, both the calls to **New_HHK**, relative to a single **Update** step, correspond to a call to the function in (HHK95) on a graph having Σ_i nodes linked by two types of edges: $\forall$-edges and $\exists$-edges (moreover the first call is stopped after only one iteration). The only substantial difference between the function in (HHK95) and **New_HHK** is that some of the operations of predecessor's set's computation are specialized for the $\forall$-edges. As two nodes are linked by a $\forall$-edge only if they are linked by an $\exists$-edge and there are $O(|\rightarrow|)$ $\exists$-edges, we can conclude, using the results in (HHK95), that each call to **New_HHK** costs $O(|S||\rightarrow|)$.

However, as the authors of (HHK95), in order to make the innermost if in the **New_HHK** achieving a total complexity of $O(|S||\rightarrow|)$ in each procedure's call, we need supposing to dispose of a special counter table. This counter table keeps track, for each couple of classes $\langle \alpha, \beta \rangle$, of the value $|post(\alpha) \cap sim(\beta)|$ requiring $O(|S|^2 \log(|S|))$ space. Exploiting a sort of global argument, we will now prove that the cost of

performing the innermost **New_HHK** if statement overall the **Partitioning Algorithm** is indeed $O(|S|^2| \rightarrow |)$, also supposing not to dispose of the just mentioned counter tables. Practically the innermost **New_HHK** if statement is executed only after a class, say β, is removed from the set of classes simulating another class, say α; its cost, supposing not to dispose of any counter table, is easily proved to be $O(|S|| \rightarrow^{-1} (\beta)|)$. Let α_k be a class in Σ_k and consider, for each iteration of the **Partitioning Algorithm** i, $\alpha_i \supset \alpha_k$ and the classes $\beta_i^1, .., \beta_i^{m_i}$ removed from $sim(\alpha_i)$ while executing the innermost for-loop in a **New_HHK** call. By definition 4.18, corollary 4.30 and lemma 4.31 the classes $\beta_1^1, .., \beta_1^{m_1}, .., \beta_k^1, .., \beta_k^{m_k}$ are mutually disjoint; hence the cost of executing the innermost **New_HHK** if statement, involving a class just recognized not able to simulate $\alpha_i \supset \alpha_k$, is $(| \rightarrow^{-1} (\beta_1^1)| + .. + | \rightarrow^{-1} (\beta_1^{m_1})| + .. + | \rightarrow^{-1} (\beta_k^1)| + .. + | \rightarrow^{-1} (\beta_k^{m_k})|)|S| = O(| \rightarrow ||S|)$. As there are $|S|$ classes in Σ_k, the innermost if statement of the function **New_HHK** takes, overall the entire **Partitioning Algorithm**, $O(|S|^2| \rightarrow |)$. □

THEOREM 4.34 (Space complexity). *Given a graph $G = (N, \rightarrow)$ and a partition pair $\langle \Sigma, P \rangle$, with P transitive, the algorithm* **Partitioning Algorithm** *computes the solution $\langle S, \preceq \rangle$ of the GCPP over them in space $O(|S|^2 + |N| \log(|S|))$.*

Proof. During each iteration of the algorithm it is necessary to consider:

- the relation $\rightarrow$;
- the relation P_i: at most $O(|\Sigma_i|^2)$ space;
- the relation which maps each node in N into the class of Σ_i to which it belongs: space $O(|N| \log |\Sigma_i|)$;
- a counters' table necessary to the function **New_HHK** in order to work in time $O(|\Sigma_i|| \rightarrow |)$ (see (HHK95)): this takes space $O(|\Sigma_i|^2)$). We observe that it is not really necessary to keep the relation $\rightarrow$ in memory: we use it only when it is necessary to provide the set of successors and predecessors of a given node.

Hence the space complexity is $O(|S|^2 + |N| \log(|S|)))$. □

4.4. Implementation and Tests

To asses the performance of the **Partitioning Algorithm** we have implemented it in Standard ML of the New Jersey and interfaced it with the Concurrency Workbench of the New Century (CWB-NC) (see (CS96)). The CWB-NC release incorporates both the simulation algorithm by Paige and Bloom (BP95) and the simulation algorithm by Cleaveland and Tan (CT01). We tested our routine and the latter mentioned procedure on some toy examples as well as on case studies included in the CWB-NC.

The CWB-NC analysis routines work on transitions systems having labelled edges. Thus, we adapted our algorithm following an approach similar to the ones used in (CT01). The **Refine** step is performed once with respect to each action, while in the **Update** step an action parameter is employed.

In the sequel we describe some of the data structures used giving several implementation details. Then, some experimental results will be presented. Further details on the implementation as well as on the tests are available at `http://www.dimi.uniud.it/~gentilin`.

We use two modifiable arrays of records (coarser_partition_table and finer_partition_table) to represent, in each iteration i of the algorithm, the partition Σ_{i-1} (to be refined) and the partition Σ_i (result of the refinement step). Each record in the finer_partition_table corresponds to a block of Σ_i and we associate to it the following fields:

- states: a doubly linked list of states representing the set of states belonging to the block;
- touched_states: a doubly linked list of states used to keep trace of the states touched while scanning the elements having transitions into a Σ_{i-1} class;
- superclass: the index in the coarser_partition_table of the Σ_{i-1} class containing the block;
- stable_blocks: a list of indexes in the coarser_partition_table allowing to represent Stable(α') within the **Refine** step.

Each record in the coarser_partition_table corresponds to a block of Σ_{i-1} and we keep the following information associated to it:

- splitted_blocks: a list of indexes in the finer_partition_table corresponding to the blocks $\alpha' \in \Sigma_i$ such that $\alpha' \subseteq \alpha$;
- greater_blocks: a list of indexes in the coarser_partition_table corresponding to the blocks $\beta \in \Sigma_{i-1}$ such that $(\alpha, \beta) \in P_{i-1}$.

We do not represent explicitly the set of states of each class $\alpha \in \Sigma_{i-1}$: this set can be retrieved by combining the doubly linked list of states of each Σ_i class contained in α. The states are integers ranging over [1..num_states]. Thus, they are used to index the table states_info maintaining, for each state, a pointer to the position in the unique doubly linked list (states or touched_states) they belong to. The relation $\rightarrow^{-1}$ is maintained, by means of an adjacency list. This allows to retrieves, for each node a, the set $\rightarrow^{-1}(a)$ in time proportional to its size.

At the beginning of each refinement step i the coarser_partition_table and the finer_partition_table represent the same partition Σ_{i-1}. After the refinement has been performed the finer_partition_table keeps Σ_i and the splitted_blocks lists in the coarser_partition_table allow to retrieve the correspondence between Σ_{i-1} and Σ_i. Hence, it is possible to construct $\exists\forall$ quotient structure and the $\exists\forall$ induced quotient structure. These are represented by means of two pairs of adjacency lists. In the adjacency list representing $\exists$ quotient structure, for example, we associate to each Σ_i class, α, the list of Σ_i classes β such that $\alpha \rightarrow^{\exists} \beta$. This allows us to retrieve, given a block α, the set of blocks β with $\alpha \rightarrow^{\exists} \beta$ in time proportional to its size. Beside the construction of the quotient structures some other operations are necessary before entering the i step of update. In particular, the partition Σ_i is copied in the coarser_partition_table. Meanwhile the relation relation induced over Σ_i by P_{i-1} is computed and stored in the greater_blocks fields of the coarser_partition_table. The remove sets used in **New_KKH** are globally represented as an array of lists of indexes in the coarser_partition_table.

Now we present the results of some tests performed to compare our implementation of the **Partitioning Algorithm** with the implementation of the algorithm by Cleaveland and Tan ((CT01)) available inside CWB-NC[4]. The tests have been executed on a Pentium III, 400 MHz PC, 256MB RAM, OS Linux Red Hat 6.2. Times are expressed in seconds.

We start with some tests on toy examples built to point out the differences between the two algorithms. Figure 11 shows the structures of two examples. In this two cases the **Partitioning Algorithm** take advantage of the fact that its time and space complexities depend on the size of the simulation quotient. In both the examples all the states are not bisimilar. In the example on the left of Figure 11, we call it Tree, each level of the tree corresponds to a simulation class and the block α_0 is simulated by all the other blocks. Hence, the size of the

[4] The implementation of the Cleaveland and Tan algorithm we use is the one with ALT table and without Path Compression (see (CT01)).

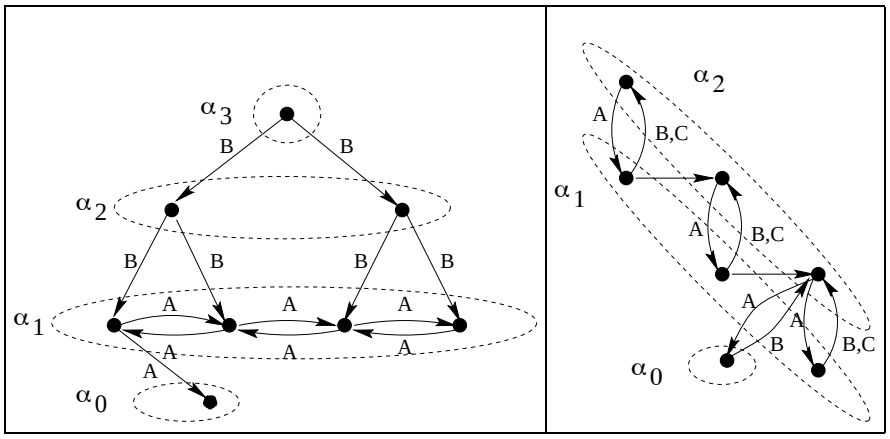

Figure 11. The Tree and the 3-classes examples.

Table I. Results of the tests on Tree (Figure 11 left).

States	**Cleaveland Tan**	**Partitioning Algorithm**
256	0.54	0.13
512	2.19	0.28
1024	8.41	0.61
2048	–	1.42
8192	–	8.53

simulation quotient is logarithmic with respect to the state space. The example on the right of Figure 11, we call it 3-classes, instead has three classes of simulation, α_0, α_1, and α_2, and α_0 is simulated by α_1. Hence, the size of the simulation quotient of 3-classes is three, while the size of the bisimulation quotient is equal to the size of the state space.

Table I (Table II, resp.) shows the results of running the two simulation procedures over graphs having the structure of Tree (3-classes, resp.) with increasing state space sizes. In the tables we use $--$ in the cases in which we run out of memory. As expected, since in both examples the simulation reduction is larger than the bisimulation one, the **Partitioning Algorithm** uses less time and space than the procedure by Cleaveland and Tan.

Notice that if in the Tree example we remove the node in the class α_0 we have that the bisimulation and the simulation quotients coincide and they are given by the levels of the tree. In Table III we show the results of the tests performed on this variant of the Tree. In this case the

Table II. Results of the tests on 3-classes (Figure 11 right).

States	**Cleaveland Tan**	**Partitioning Algorithm**
500	0.14	0.01
1000	12.56	0.08
1500	27.65	0.25
2000	–	0.32
10000	–	2.18

Table III. Results of the tests on a variant of Tree.

States	**Cleaveland Tan**	**Partitioning Algorithm**
256	0.06	0.13
512	0.16	0.3
1024	0.34	0.71
2048	1.16	1.54
8192	5.98	9.01

Cleaveland and Tan algorithm takes advantage of the large (maximal) bisimulation reduction and performs better than our algorithm.

In Tables IV and V we report the result of a test performed using a benchmark taken from CWB-NC. In particular, Table IV shows some information about the structures of different models of the alternating-bit protocol included in the CWB-NC release. In Table V the times resulting from minimizing the systems are reported. This last example shows the space efficiency of the **Partitioning Algorithm** on a more concrete example. The time performances of the **Partitioning Algorithm** may be surprising unless one considers that the $O(|S^2|| \rightarrow |)$ time complexity is reached only in cases in which in each iteration only few splits are performed and $|S|$ iterations are needed to get to the simulation quotient.

5. Conclusions

In this paper we discussed the notions of bisimulation and simulation, and the role they play as graph reduction procedures with special emphasis on their use in verification. The search for efficient procedures determining the bisimulation reduction is shown to be solvable passing to an equivalent coarsest partition problem first, which is then

Table IV. Characteristics of four bit-alternating-protocol models included in the CWB-NC.

	States	Transitions	Bisim Classes	Sim classes
ABP-lossy	57	130	15	14
ABP-safe	49	74	18	17
Two-link-netw	1589	6819	197	147
Three-link-netw	44431	280456	2745	1470

Table V. Tests over the systems in Table IV.

	Cleaveland Tan	**Partitioning Algorithm**
ABP-lossy	0.05	0.04
ABP-safe	0.04	0.03
Two-link-netw	7.53	5.16
Three-link-netw	–	1336.24

observed to be nothing but an equality problem on non-well-founded sets. Moving to simulation, the circle of ideas discussed is also shown to be useful in the study and design of fast simulation algorithms and heuristics especially developed to operate in situations in which strong space constraint are present. Moreover, the use of the $\forall\exists$-structure and the definition of a coarsest partition problems on them, seem to be a methodology with some potential in all those situations in which a fix-point in the lattice of equivalence relations is to be computed.

We plan to work on a symbolic version of our algorithms, a version which is naturally suggested by the fact that the discussed procedure work on sets of nodes (the classes of the partitions). Finally, an attempt to combine the negative and positive strategies to solve the coarsest partition problem for simulation presented here (as, in the case of bisimulation, has been done in (DPP01)) is also under study.

References

P. Aczel. *Non-well-founded sets*, volume 14 of *CSLI Lecture Notes*. Stanford University Press, 1988.

A. Bouali and R. de Simone. Symbolic bisimulation minimization. In G. von Bochmann and D. K. Probst, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'92)*, volume 663 of *LNCS*, pages 96–108. Springer, 1992.

J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam, Instituut voor Logica en Grondslagenonderzoek van Exacte Wetenschappen, 1976.

A. Bouajjani, J.C. Fernandez, and N. Halbwachs. Minimal model generation. In E. Clarke and R. Kurshan, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'90)*, volume 531 of *LNCS*, pages 197–203. Springer, 1990.

D. Bustan and O. Grumberg. Simulation based minimization. In D.A. McAllester, editor, *Proc. of Int. Conference on Automated Deduction (CADE'00)*, volume 1831 of *LNCS*, pages 255–270. Springer, 2000.

R. Bloem, H. N. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. In W. A. Hunt Jr. and S. D. Johnson, editors, *Proc. of Int. Conference on Formal Methods in Computer-Aided Design (FMCAD'00)*, volume 1954 of *LNCS*, pages 37–54. Springer, 2000.

B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 1989.

A. Bouali. XEVE, an ESTEREL verification environment. In A. J. Hu and M. Y. Vardi, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 500–504. Springer, 1998.

B. Bloom and R. Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Science of Computer Programming*, 24(3):189–220, June 1995.

R. E. Bryant. Symbolic manipulation of boolean functions using a graphical representation. In *Proc. of Design Automation Conference (DAC'85)*, 1985.

E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using brancing time temporal logic. In *Proc. of Workshop on Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1982.

E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 1999.

R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.

R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In K.G. Larsen and A. Skou, editors, *Proc of Int. Conference on Computer Aided Verification (CAV'91)*, volume 575 of *LNCS*, pages 48–58. Springer, 1992.

R. Cleaveland and S. Sims. The NCSU concurrency workbench. In R. Alur and T. A. Henzinger, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 394–397. Springer, 1996.

R. Cleaveland and L. Tan. Simulation revised. In T. Margaria and W. Yi, editors, *Proc. of Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *LNCS*, pages 480–495. Springer, 2001.

D. Dams, R. Gerth, and O. Grumberg. Generation of reduced models for checking fragments of CTL. In C. Courcoubetis, editor, *Proc. of Int. Conference on Computer Aided Verification (CAV'93)*, volume 697 of *LNCS*, pages 479–490. Springer, 1993.

D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, March 1997.

A. Dovier, R. Gentilini, C. Piazza, and A. Policriti. Rank-based symbolic bisimulation (and model checking). In Ruy J. Guerra B. de Queiroz, editor, *Proc.*

of Workshop on Language, Logic, Information, and Computation (Wollic'02), volume 67 of *ENTCS*, pages 167–184. Elsevier Science, 2002.

A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 79–90. Springer, 2001.

R. Focardi and R. Gorrieri. The compositional security checker: a tool for the verification of information flow security properties. *IEEE Transaction on Software Engineering*, 23(9):550–571, 1997.

J. C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A protocol validation and verification toolbox. In R. Alur and T. A. Henzinger, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 437–440. Springer, 1996.

M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore di Pisa, Cl. Sc.*, IV(10):493–522, 1983.

K. Fisler and M. Y. Vardi. Bisimulation and model checking. In L. Pierre and T. Kropf, editors, *Proc. of Correct Hardware Design and Verification Methods (CHARME'99)*, volume 1703 of *LNCS*, pages 338–341. Springer, 1999.

O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and systems*, 16(3):843–871, May 1994.

R. Gentilini, C. Piazza, and A. Policriti. Simulation as coarsest partition problem. In J. P. Katoen and P. Stevens, editors, *Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *LNCS*, pages 415–430. Springer, 2002.

R. Gentilini, C. Piazza, and A. Policriti. Simulation reduction as constraint. In M. Falaschi, editor, *Proc. of Workshop on Functional and Constraint Logic Programming (WFLP'02)*, pages 59–72. Research Report UDMI/18/2002/RR, 2002.

R. Gentilini, C. Piazza, and A. Policriti. Computing strongly connected components in a linear number of symbolic steps. In *Proc. of Int. Symposium on Discrete Algorithms (SODA'03)*, ACM, 2003. To appear.

D. Gries. Describing an algorithm by hopcroft. *Acta Informatica*, 2:97–109, 1973.

M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of Symposium on Foundations of Computer Science (FOCS'95)*, pages 453–462. IEEE Computer Society Press, 1995.

J.E. Hopcroft. An *nlogn* algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations, Ed. by Zvi Kohavi and Azaria Paz*, pages 189–196. Academic Press, 1971.

T. Knuutila. Re-describing an algorithm by hopcroft. *Theoretical Computer Science*, 250:333–363, 2001.

P. C. Kannellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

D. Lee and M. Yannakakis. Online minimization of transition systems. In *Proc. of ACM Symposium on Theory of Computing (STOC'92)*, pages 264–274. ACM Press, 1992.

K. L. McMillan. *Symbolic model checking: an approach to the state explosion problem*. Kluwer Academic Publishers, 1993.

R. Milner. An algebraic definition of simulation between programs. In *Proc. of Int. Joint Conference on Artificial Intelligence (IJCAI'71)*, pages 481–489. Morgan Kaufmann, 1971.

R. Milner. A calculus of communicating systems. In G. Goos and J. Hartmanis, editors, *Lecture Notes on Computer Science*, volume 92. Springer, 1980.

D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. of Int. Conference on Theorical Computer Science (TCS'81)*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

C. Piazza. *Computing in Non Standard Set Theories*. Cs, Department of Computer Science, University of Udine, 2002. Electronically available from `http://www.dimi.uniud.it/~piazza`.

C. Piazza and A. Policriti. Ackermann encoding, bisimulations, and OBDD's. In M. Leuschel, A. Podelski, C.R. Ramakrishnan, and U. Ultes-Nitsche, editors, *Proc. of Int. Workshop on Verification and Computational Logic (VCL'01)*, volume DSSE-TR-2001-3 of *Southampton University Technical Report*, pages 43–53. ACM Digital Library, 2001.

R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

R. Paige, R. E. Tarjan, and R. Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40(1):67–84, 1985.

W. R. Roscoe. *A Classical Mind: Essays in Honour of C.A.R. Hoare*, chapter Model Checking CSP. Prentice Hall, 1994.

6. Appendix: Proofs

Proof of Lemma 2.8. The identity relation I over N is always a simulation relation over G, hence, since N is finite, the set $\mathcal{S}$ of the simulation relations over G is finite and not empty.

We prove that if $\leq_1$ and $\leq_2$ are simulation relations over G, then $\leq = \leq_1 \cup \leq_2$ is a simulation relation over G. If $a \leq b$, then $a \leq_1 b$ or $a \leq_2 b$: we can safely assume that $a \leq_1 b$. Since $\leq_1$ is a simulation relation we obtain that $[a]_\Sigma = [b]_\Sigma$ and $\forall c \in N(a \to c \Rightarrow \exists d \in N(c \leq_1 d \wedge b \to d))$. Since $\leq = \leq_1 \cup \leq_2$ we obtain that $[a]_\Sigma = [b]_\Sigma$ and $\forall c \in N(a \to c \Rightarrow \exists d \in N(c \leq d \wedge b \to d))$, which means that $\leq$ is a simulation.

Consider the relation

$$\leq_s = \bigcup_{\leq \in \mathcal{S}} \leq .$$

Since $\mathcal{S}$ is finite and not empty $\leq_s$ is correctly defined. We have proved that union preserves *be a simulation relation*, hence $\leq_s$ is the unique maximal simulation relation over G.

Since I is a simulation relation $\leq_s$ is reflexive. In order to prove that $\leq_s$ is transitive we prove that if $\leq$ is a simulation relation, then its transitive closure $\leq^+$ is a simulation relation. If $a \leq^+ b$, then there exists $a_1, \ldots, a_k$ in N such that $a \leq a_1 \leq \ldots \leq a_k \leq b$. Hence we obtain that $[a] = [a_1] = \ldots = [a_k] = [b]$. If $a \to c$, then there exists c_1 such that $a_1 \to c_1$ and $c \leq c_1$. Applying the same argument to a_i, c_i and a_{i+1}, for $i = 1, \ldots, k-1$, and to a_k, c_k and b we obtain a chain of

the form $c \leq c_1 \leq \ldots \leq d$. Hence, since $\leq^+$ is the transitive closure of $\leq$ we have that $c \leq^+ d$, i.e. our thesis. □

Proof of Proposition 3.3.

(*i*) We prove that $\asymp_\Pi$ is a bisimulation on G. Since Π refines Σ, it holds that if $a \asymp_\Pi b$, then a and b belong to the same class in Σ. Let a, b be such that $a \asymp_\Pi b$. This means that there is a class $\alpha \in \Pi$ such that $a \in \alpha$ and $b \in \alpha$. Assume there is $c \in N$ such that $a \to c$. Let γ be the class such that $c \in \gamma$. Since Π is stable with respect to $\to$ and $\to^{-1}(\gamma)$ is not empty (it contains a), this means that $\alpha \subseteq \to^{-1}(\gamma)$. Thus, $b \in \to^{-1}(\gamma)$, and this implies that there is $d \in \gamma$ such that $b \to d$. By definition of $\asymp_\Pi$, $c \asymp_\Pi d$. Similarly we can prove that if $a \asymp_\Pi b$ and $b \to d$, then there exists c such that $c \asymp_\Pi d$ and $a \to c$.

(*ii*) By contradiction, assume that $\Pi_\asymp$ is not stable with respect to $\to$. This means that there are blocks α and γ and two nodes a, b such that

$$a \in \alpha \setminus \to^{-1}(\gamma) \wedge b \in \alpha \cap \to^{-1}(\gamma)$$

This implies that there is a node $d \in \gamma$ such that $b \to d$ but no node c bisimilar to d (i.e., in γ) can be reached by an edge from a. Thus, $\neg a \asymp b$.

□

Proof of Proposition 3.10. Here with minimum graph bisimilar to G we mean that if $G' = (N', \to', \ell')$ is bisimilar to G, then $|N'| \geq N_{\equiv_b}$.

It can be easily proved that the relation $\forall a \in N(aB[a]_b)$ is a bisimulation between G and $G/\equiv_b$.

Let $G_1 = (N_1, \to_1, \ell_1)$ be a graph and B_1 be a bisimulation between G and G_1. The relation B_1' defined as

$$\forall a, b \in N(aB_1'b \Leftrightarrow \exists k \in N_1(aB_1k \wedge bB_1k)).$$

is easily seen to be a bisimulation. Since B_1' is included in $\equiv_b$, it holds that $|N/\equiv_b| \leq N_1$. □

Proof of Lemma 4.3. Let $r(\leq^{G'}) = \leq^G$. We have to prove that $\leq^G$ is a simulation over G. If $a \leq^G b$ and $a \in \alpha \in \Sigma$, then since $\leq^{G'}$ is a simulation over G', $a \to' \ell_\alpha$, and since $[\ell_\alpha]_{\Sigma'} = \{\ell_\alpha\}$, it must be $b \to' \ell_\alpha$, i.e. $b \in \alpha$. If $a \leq^G b$ and $a \to c$, then since $\leq^{G'}$ is a simulation over G', there must exist $d \in N'$ such that $c \leq^{G'} d$ and $b \to' d$. Since $c \in N$ and

in Σ' the set N is a class, from $c \leq^{G'} d$, we obtain that $d \in N$. Hence there exists d such that $c \leq^{G} d$ and $b \to d$.

Let $e(\leq^{G}) = \leq^{G'}$. We have to prove that $\leq^{G'}$ is a simulation over G. If $a \leq^{G'} b$ and $n \in L$, then the case is trivial since it must be that $b = a$. If $a \leq^{G'} b$ and $n \in N$, then we have that $b \in N$. If $a \to' c$ and $c \in L$, then from the fact that $\leq^{G}$ is a simulation, we obtain that $c \leq^{G'} c$ and $b \to' c$. If $a \to' c$ and $c \in N$, then let d be the element of N such that $c \leq^{G} d$ and $b \to d$, we have that $c \leq^{G'} d$ and $b \to' d$, which concludes this case.

The rest of the thesis follows easily from the definitions of r and e. □

Proof of Lemma 4.14. (1). We prove that if $\langle \Pi, Q \rangle$ belongs to S_{GP}, then $\leq_{\langle \Pi, Q \rangle}$ is a simulation over G'.
If $a \leq_{\langle \Pi, Q \rangle} b$, then only two cases are possible:

- a and b are elements of N;
- $a = b$ and a is an element of L.

Hence in both cases we have that they belongs to the same class in Σ'. The second case is trivial. In the first case if c is such that $a \to' c$, then it can either be $c \in L$ or $c \in N$.
If $c \in L$, from $a \to' c$ we have that $a \in \alpha \in \Sigma$, $c = \ell_\gamma$ and $(\gamma, \alpha) \in P^+$. From the fact that $a \leq_{\langle \Pi, Q \rangle} b$ we have that $a \in \alpha' \in \Pi$ and $b \in \beta' \in \Pi$ and $(\alpha', \beta') \in Q$. Hence we have that $b \in \beta \in \Sigma$ and $(\alpha, \beta) \in P^+$. From $(\gamma, \alpha) \in P^+$ and $(\alpha, \beta) \in P^+$ we obtain $(\gamma, \beta) \in P^+$, hence $c \leq_{\langle \Pi, Q \rangle} c$ and $b \to' c$.
If $c \in N$, then we have that $a \in \alpha' \in \Pi$, $c \in \gamma' \in \Pi$, $b \in \beta' \in \Pi$, $\alpha' \to_\exists \gamma'$, and $(\alpha', \beta') \in Q$ (since $a \leq_{\langle \Pi, Q \rangle} b$). Hence, since $\langle \Pi, Q \rangle$ is stable, there exists $\delta' \in \Pi$ such that $(\gamma', \delta') \in Q$ and $\beta' \to_\forall \delta'$. Let $d \in \delta'$ be such that $b \to d$. Since $c \leq_{\langle \Pi, Q \rangle} d$, the thesis follows.

It is immediate to prove that if the relation $\leq$ is a simulation over G', then $\langle \Pi_s, Q_s \rangle$ is a partition pair, i.e. Q_s is reflexive and acyclic.

(2). We prove that if $\leq$ is a simulation over G', then $\langle \Pi_s, Q_s \rangle$ is a stable refinement of $\langle \Sigma, P \rangle$.
Observe that Q_s is acyclic.
Let α', β', γ' in Π_s be such that $\alpha' \to_\exists \gamma'$ and $(\alpha', \beta') \in Q_s$. We have to prove that there exists $\delta' \in \Pi_s$ such that $(\gamma', \delta') \in Q_s$ and $\beta' \to_\forall \delta'$.
From $\alpha' \to_\exists \gamma'$ we obtain that there exist $a \in \alpha'$ and $c \in \gamma'$ such that $a \to' c$. From $(\alpha', \beta') \in Q_s$ we obtain that $a \leq^* b$ for all $a \in \alpha'$ and $b \in \beta'$. Hence, since $\leq^*$ is a simulation, we have that for all $b \in \beta'$ there exists d_b such that $b \to' d_b$ and $c \leq^* d_b$ (hence $d_b \in N$). Let $b_1, \ldots, b_t$ be an ordering of the elements of β'.

We inductively define as follows the lists $L^1, .., L^n, ..$ of elements related through $\rightarrow'$ with $b_1, \ldots, b_t$:

- $L^1 = d_1^1, \ldots, d_t^1$
  where d_1^1 is such that $b_1 \rightarrow' d_1^1$ and $c \leq^* d_1^1$ and, for $j = 2, \ldots, t$, d_j^1 is such that $b_j \rightarrow' d_j^1$ and $d_{j-1}^1 \leq^* d_j^1$. Notice that L^1 is correctly defined (i.e $d_1^1, \ldots, d_t^1$ always exist) since $\leq^*$ is a simulation, $a \leq^* b \wedge a \rightarrow' c$ and, for $j = 2, \ldots, t$, $b_{j-1} \leq^* b_j$.

- $L^{h+1} = d_1^{h+1}, \ldots, d_t^{h+1}$
  where d_1^{h+1} is such that $b_1 \rightarrow' d_1^{h+1}$ and $d_t^h \leq^* d_1^{h+1}$ and, for $j = 2, \ldots, t$, d_j^{h+1} is such that $b_j \rightarrow' d_j^{h+1}$ and $d_{j-1}^{h+1} \leq^* d_j^{h+1}$.
  L^{h+1} is correctly defined (i.e $d_1^{h+1}, \ldots, d_t^{h+1}$ always exist) since $b_1, \ldots, b_t$ is an ordering of the elements of the class β' and $\leq^*$ is a simulation.

By concatenation of the lists $L^1, L^2, \ldots$ we obtain an infinite chain of elements $d_1^1, \ldots, d_t^1, d_1^2, \ldots, d_t^2, .., d_1^n, \ldots, d_t^n, ..$ such that

1. for all $k > 0$ and for $j = 1, \ldots, t$ $b_j^k \rightarrow' d_j^k$
2. for all $k > 0$ and for $j = 2, \ldots, t$, $d_{j-1}^k \leq^* d_j^k$
3. $c \leq^* d_1^1$ and for all $k > 0$ $d_t^{k-1} \leq^* d_1^k$

Since the set of the successors of b_1 is a finite set we have that there exist v, u such that $d_1^v = d_1^u = d_1^*$ and $v < u$. Hence we obtain a prefix of the chain of the form $d_1^1 \ldots \leq^* d_1^* \leq^* \ldots \leq^* d_1^*$ with $c \leq^* d_1^1$. This means that all the elements of this succession between the two occurrences of d_1^* belongs to the same class $\delta' \in \Pi$. Moreover all the elements β' $(b_1, \ldots, b_t)$ have a successor in δ' and obviously $(\gamma', \delta') \in Q_s$, i.e. the thesis.
(3). Let $\leq_1 = f_1(\langle \Pi_1, Q_1 \rangle)$ and $\leq_2 = f_1(\langle \Pi_2, Q_2 \rangle)$. We assume that $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$ and we have to prove that $\leq_1 \subseteq \leq_2$.
If $a \leq_1 b$, then $a \in \alpha_1 \in \Pi_1$, $b \in \beta_1 \in \Pi_1$ and $(\alpha_1, \beta_1) \in Q_1$, hence $a \in \alpha_2 \in \Pi_2$, $b \in \beta_2 \in \Pi_2$ and $(\alpha_2, \beta_2) \in Q_2$, from which $a \leq_1 b$.
(4). Let $\langle \Pi_1, Q_1 \rangle = f_2(\leq_1)$ and $\langle \Pi_2, Q_2 \rangle = f_2(\leq_2)$. We assume that $\leq_1 \subseteq \leq_2$ and we prove that $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$.
From $\leq_1 \subseteq \leq_2$ we have $\leq_1^* \subseteq \leq_2^*$.
If $a, b \in \alpha_1 \in \Pi_1$, then $a \leq_1^* b \leq_1^* a$, hence $a \leq_2^* b \leq_2^* a$, from which $a, b \in \alpha_2 \in \Pi_2$, i.e. Π_1 is finer than Π_2.
If $(\alpha_1, \beta_1) \in Q_1$, then since Π_1 is finer than Π_2 there exists $\alpha_2, \beta_2 \in \Pi_2$ such that $\alpha_1 \subseteq \alpha_2$ and $\beta_1 \subseteq \beta_2$. We have to prove that $(\alpha_2, \beta_2) \in Q_2$. Let $a_1 \in \alpha_1$ and $b_1 \in \beta_1$ be such that $a_1 \leq_1^* b_1$. It holds that $a_1 \leq_2^* b_1$, $a_1 \in \alpha_2$ and $b_1 \in \beta_2$, hence $(\alpha_2, \beta_2) \in Q_2$.

(5). If $a \leq b$, then $a \in \alpha \in \Pi_s$, $b \in \beta \in \Pi_s$ and $(\alpha, \beta) \in Q_s$. Hence we obtain $a \leq_{\langle \Pi_s, Q_s \rangle} b$.

(6). If $a, b \in \alpha \in \Pi$, then, since Q is reflexive, it holds $a \leq_{\langle \Pi, Q \rangle} b \leq_{\langle \Pi, Q \rangle} a$, hence $a, b \in \alpha' \in \Pi_{\langle \Pi, Q \rangle}$, i.e. Π is finer than $\Pi_{\langle \Pi, Q \rangle}$. Let $(\alpha, \beta) \in Q$, consider $\alpha', \beta' \in \Pi_{\langle \Pi, Q \rangle}$ such that $\alpha \subseteq \alpha'$ and $\beta \subseteq \beta'$. Let $a_1 \in \alpha$ and $b_1 \in \beta$. It holds that $a_1 \leq_{\langle \Pi, Q \rangle} b_1$, $a_1 \in \alpha'$, and $b_1 \in \beta'$, hence $(\alpha', \beta') \in Q_{\langle \Pi, Q \rangle}$. □

Proof of Theorem 4.15. Let $\leq_s$ be the maximal simulation over G'. From Lemma 4.14(2) we have that $\langle \Pi_m, Q_m \rangle = f_2(\leq_s)$ is a stable refinement of $\langle \Sigma, P \rangle$. We prove that if $\langle \Pi, Q \rangle$ is another stable refinement, then $\langle \Pi, Q \rangle \sqsubseteq \langle \Pi_m, Q_m \rangle$. From Lemma 4.14(6) we know that $\langle \Pi, Q \rangle \sqsubseteq f_2(f_1(\langle \Pi, Q \rangle))$. Since $f_1(\langle \Pi, Q \rangle)$ is a simulation over G' we also know that $f_1(\langle \Pi, Q \rangle) \subseteq \leq_s^m$. Hence, using Lemma 4.14(4) we obtain that $\langle \Pi, Q \rangle \sqsubseteq f_2(f_1(\langle \Pi, Q \rangle)) \sqsubseteq f_2(\leq_s^m) = \langle \Pi_m, Q_m \rangle$. As for the uniqueness, assume $\langle \Pi', Q' \rangle \neq \langle \Pi_m, Q_m \rangle$ is another solution. Then, $\langle \Pi_m, Q_m \rangle \sqsubseteq \langle \Pi', Q' \rangle$. Using Definition 4.13 of f_1 we immediately deduce $f_1(\langle \Pi', Q' \rangle) \neq f_1(\langle \Pi_m, Q_m \rangle)$. By Lemma 4.14(3) and 4.14(5) we then obtain $\leq_s = f_1(f_2(\leq_s)) \subset f_1(\langle \Pi', Q' \rangle)$ which is a contradiction. □

Proof of Lemma 4.19. Consider the partition Π_1 defined as $\Pi_1 = \{\{a\} \mid a \in N\}$. Clearly Π_1 is finer than Σ.
Π_1 satisfies all the conditions in $(1)\sigma$ (but it can be the case that it is not a coarsest), since P is reflexive.
Let Π be a partition such that:

- Π is finer than Σ;
- Π_1 is finer than Π;
- Π satisfies all the conditions in $(1)\sigma$;
- if Π is finer than Π_2 and Π_2 is finer than Σ, then Π_2 does not satisfy the conditions in $(1)\sigma$.

Notice that there exists at least a partition Π which satisfies all these conditions because there are only a finite number of partitions.

Consider $Q_1 = \{(\alpha, \alpha) \mid \alpha \in \Pi\}$ over Π.
Clearly Q_1 satisfies all the conditions in $(2)\sigma$, but it can be the case that it is not maximal.
Let Q be a relation over Π such that:

- $Q_1 \subseteq Q \subseteq P(\Pi)$;
- Q_1 is acyclic;

- Q satisfies all the conditions in $(2)\sigma$;
- if Q_2 is such that $Q \subseteq Q_2 \subseteq P(\Pi)$, then Q_2 does not satisfy the conditions in $(2)\sigma$.

There exists at least a relation Q which satisfies all these conditions because there are only a finite number of relations. □

Proof of Theorem 4.20. The previous lemma states that there always exists a maximal pair $\langle \Pi, Q \rangle$ which satisfies the conditions in Definition 4.18; we have to prove that this maximal pair is unique.
Let Sort be a topological sorting of the elements in Σ with respect to P. Consider Π defined in Figure 12.

$\Pi := \Sigma$;
while Sort $\neq \emptyset$ do
    $\gamma := dequeue(\mathsf{Sort})$;
    for each $\alpha \in \Pi$ do
        if $\alpha \rightarrow_\exists \gamma \wedge \forall\delta((\gamma,\delta) \in P \Rightarrow \neg\alpha \rightarrow_\forall \delta)$ then
            replace α with $\alpha\cap \rightarrow^{-1} (\gamma)$ and $\alpha\setminus \rightarrow^{-1} (\gamma)$ in Π

Figure 12.

Notice that Π does not depend on the topological sorting we have chosen.
It is clear that Π satisfies (1σ).
Let us assume by contradiction that there exists Π_1 such that Π_1 satisfies (1σ) and Π_1 is not finer than Π.
This means that there exist $a, b \in N$, $\alpha \in \Pi_1$ and $\alpha_a, \alpha_b \in \Pi$ such that $a, b \in \alpha$, $a \in \alpha_a$, $b \in \alpha_b$ and $\alpha_a \neq \alpha_b$.
Consider the class α' which was obtained during the construction of Π and which was the smallest class obtained during the construction such that $\alpha \subseteq \alpha'$.
Let γ be the class of Σ which first split α' into two parts.
Notice that for all δ such that $(\gamma, \delta) \in P$ and $\delta \neq \gamma$ it holds $\neg\alpha' \rightarrow_\exists \delta$ (since all these δ's have already been extracted from Sort).
Hence it holds that for all δ such that $(\gamma, \delta) \in P$ and $\delta \neq \gamma$ $\neg\alpha \rightarrow_\forall \delta$, moreover, since α is not a subset of one of the two part of α', $\neg\alpha \rightarrow_\forall \gamma$ and $\alpha \rightarrow_\exists \gamma$. This is in contradiction with the fact that Π_1 satisfies (1σ).

Similarly it is possible to prove that there exists a unique maximal relation Q over Π which satisfies (2σ).

It is possible to obtain Q from $P(\Pi)$ by removing all the pairs which do not fulfill the condition, until a fix point is reached. It is immediate to

prove that such a fix point is reflexive. In fact, being $\langle \Sigma, P \rangle$ a partition pair over G, we have that P and hence $P(\Pi)$ are reflexive. As each pair belonging to $\{(\alpha, \alpha) \mid \alpha \in \Pi\}$ respect both condition (b) in (2σ) and condition (c) in (2σ) we can conclude that Q is reflexive.

As far as the acyclicity of Q is concerned consider the relation Q_b on Π obtained from $P(\Pi)$ by removing those pairs which don't respect condition (b) in (2σ); we will prove that Q_b is acyclic. Being Q contained in Q_b we will obtain, as a byproduct, the acyclicity of Q.

Assume by contradiction that Q_b is cyclic, i.e. there exist $\alpha_1, \alpha_2, .., \alpha_n \in \Pi$ with $\alpha_i \neq \alpha_j$ for $1 \leq i, j \leq n, i \neq j$ such that $(\alpha_1, \alpha_2) \in Q_b \wedge .. \wedge ..(\alpha_{n-1}, \alpha_n) \in Q_b \wedge (\alpha_n, \alpha_1) \in Q_b$. Since P is acyclic and $Q \subseteq P(\Pi)$ there must exists a class $\alpha \in \Sigma$ such that $\alpha_1, \ldots, \alpha_n \subseteq \alpha$. Let α be the class in Σ such that $\alpha_i \subseteq \alpha$ for $1 \leq i \leq n$. We have just proved that Π i.e the coarsest partition finer than Σ which respect condition (a) in (2σ) can be obtained using the procedure in Figure 12, where *Sort* is a topological sorting of the elements in Σ with respect to P. As, for $1 \leq i, j \leq n \wedge i \neq j$, we have $\alpha_i \neq \alpha_j$ and $\alpha_i \subseteq \alpha \in \Sigma$, by the above pseudo-code we deduce that there exist $\gamma \in \Sigma$, a permutation of $\alpha_1, .., \alpha_n$, say $\alpha'_1, .., \alpha'_n$, and an index $1 \leq k \leq n-1$ such that

$$\alpha'_1 \rightarrow_\forall \gamma \wedge .. \wedge \alpha'_k \rightarrow_\forall \gamma \wedge \nexists \delta \in \Sigma((\gamma, \delta) \in P \wedge (\alpha'_{k+1} \rightarrow_\exists \delta \vee .. \vee \alpha'_n \rightarrow_\exists \delta)) \tag{1}$$

Let's use the letters S and N to refer respectively to $\alpha'_1 \cup .. \cup \alpha'_k$ and $\alpha'_{k+1} \cup .. \cup \alpha'_n$. As $\alpha'_1, .., \alpha'_n$ is a permutation of $\alpha_1, .., \alpha_n$ and we have supposed that $(\alpha_1, \alpha_2) \in Q_b \wedge .. \wedge ..(\alpha_{n-1}, \alpha_n) \in Q_b \wedge (\alpha_n, \alpha_1) \in Q_b$ we have that there exist two indexes $1 \leq t, s \leq n$ such that $s = t+1 \vee (t = n \wedge s = 1)$ and $\alpha_t \in S \wedge \alpha_s \in N$. From $\alpha_t \in S$ we obtain that $\alpha_t \rightarrow_\forall \gamma$; since $(\alpha_t, \alpha_s) \in Q_b$ and Q_b respect condition (b) in (2σ) we have that $\alpha_s \rightarrow_\exists \delta \wedge (\gamma, \delta) \in P$ and $\alpha_s \in N$ which contradicts the condition in Equation (1). □

Proof of Lemma 4.21. From Corollary 4.16 we have that $\preceq$ is transitive.
We have to prove that if $\langle S, \preceq \rangle \sqsubseteq \langle \Pi, Q \rangle$, then $\langle S, \preceq \rangle \sqsubseteq \sigma(\langle \Pi, Q \rangle) = \langle \Pi', Q' \rangle$ i.e that

1. S is a partition finer than Π'
2. The partial order (on S) $\preceq$ is included in the relation induced on S by Q' (which is an acyclic and reflessive relation on Π')

1. If we prove that $\langle S, \preceq \rangle$ satisfies $(1\sigma)(a)$ (over $\langle \Pi, Q \rangle$) we can then conclude, by theorem 4.20, that S is comparable with with Π' and finer then Π' i.e our thesis. So let's start proving that $\langle S, \preceq \rangle$ satisfies

$(1\sigma)(a)$ (over $\langle \Pi, Q\rangle$)).
If $\beta \in S$ and $\alpha \in \Pi$ are such that $\beta \rightarrow_\exists \alpha$, then there exists $\alpha' \in S$ such that $\alpha' \subseteq \alpha$ and $\beta \rightarrow_\exists \alpha'$. Since $\langle S, \preceq\rangle$ is the solution of a GCPP it holds that there exists $\delta' \in S$ such that $(\alpha', \delta') \in \preceq$ and $\beta \rightarrow_\forall \delta'$. Hence since $\preceq \subseteq Q(S)$ it holds that $\delta' \subseteq \delta \in \Pi$, $(\alpha, \delta) \in Q$, and $\beta \rightarrow_\forall \delta$, i.e. (1σ) is satisfied.

2. we have to prove that the partial order (on S) $\preceq$ is included in the relation induced on S by Q', where $\langle \Pi', Q'\rangle = \sigma(\langle \Pi, Q\rangle)$. Notice that as a by-product of 1. we have that the partition S is finer than Π' which is finer than Π.
We will use the notation $\alpha_P, \beta_P..$ to denote the elements of a partition P.
Suppose by contradiction that there exists $\overline{\alpha}_S, \overline{\beta}_S, \overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'}$ such that $\overline{\alpha}_S \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_S \subseteq \overline{\beta}_{\Pi'}$ and $(\overline{\alpha}_S \preceq \overline{\beta}_S \wedge (\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'}) \notin Q')$. Without losing generality we can assume that the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is the first one which is assumed not to bee in Q' and is such that there exist $\overline{\alpha}_S \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_S \subseteq \overline{\beta}_{\Pi'}$ with $\overline{\alpha}_S \preceq \overline{\beta}_S$. We have to consider 3 cases:

   a) the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is not in the relation induced on Π' by Q;
   b) the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ has been removed from the relation induced on Π' by Q in order to satisfy condition (b) in (2σ);
   c) the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ has been removed from Q' in order to satisfy condition (c) in (2σ).

   a) Consider the classes of Π, $\overline{\alpha}_\Pi$ and $\overline{\beta}_\Pi$ such that $\overline{\alpha}_\Pi \supseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_\Pi \supseteq \overline{\beta}_{\Pi'}$; we immediately deduce that $\overline{\alpha}_S \supseteq \overline{\alpha}_\Pi, \overline{\beta}_S \supseteq \overline{\beta}_\Pi$ and $(\overline{\alpha}_S \preceq \overline{\beta}_S \wedge (\overline{\alpha}_\Pi, \overline{\beta}_\Pi) \notin Q,)$ which contradicts our assumption that $\langle S, \preceq\rangle \sqsubseteq \langle \Pi, Q\rangle$.
   b) Recall that since $(\langle S, \preceq\rangle)$ is the solution of the GCPP it holds that $(\langle S, \preceq\rangle)$ is stable i.e

$$\forall \alpha_S, \beta_S, \gamma_S (\alpha_S \preceq \beta_S \wedge \alpha_S \rightarrow_\exists \gamma_S \Rightarrow \exists \delta_S (\gamma_S \preceq \delta_S \wedge \beta_S \rightarrow_\forall \delta_S)).$$

   If the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is assumed not to bee in Q' because of condition (b) in (2σ), then there exists a class of Π, γ_Π, such that:

$$\overline{\alpha}_{\Pi'} \rightarrow_\forall \gamma_\Pi \wedge \neg \exists \delta_\Pi ((\gamma_\Pi, \delta_\Pi) \in Q \wedge \overline{\beta}_{\Pi'} \rightarrow_\exists \delta_\Pi) \qquad (2)$$

   From $\overline{\alpha}_{\Pi'} \rightarrow_\forall \gamma_\Pi$ we obtain that that for every class of S, α_S which is included[5] in $\alpha_{\Pi'}$ there exists a class of S included in

[5] From 1. we have that the partition S is finer than Π' which is finer then Π.

γ_Π, γ_S, such that $\alpha_S \rightarrow_\exists \gamma_S$. Hence we have that there is a class of S, say γ_S^*, included in γ_Π and such that $\overline{\alpha}_S \rightarrow_\exists \gamma_S^*$. From the second conjunct in assertion (2), $\gamma_S^* \subseteq \gamma_\Pi$, $\overline{\beta}_S \subseteq \overline{\beta}_{\Pi'}$ and from our assumption that $\preceq$ is included in the relation on S induced by Q we deduce that

$$\neg\exists\delta_S(\gamma_S^* \preceq \delta_S \wedge \overline{\beta}_S \rightarrow_\exists \delta_S)$$

and hence that

$$\overline{\alpha}_S \preceq \overline{\beta}_S \wedge \overline{\alpha}_S \rightarrow_\exists \gamma_S^* \wedge \neg\exists\delta_S(\gamma_S^* \preceq \delta_S \wedge \overline{\beta}_S \rightarrow_\forall \delta_S)$$

i.e $\langle S, \preceq\rangle$ is not stable, which is a contradiction.

c) Recall that we assumed that the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is the first one which is not in Q' and is such that there exist $\overline{\alpha}_S \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_S \subseteq \overline{\beta}_{\Pi'}$ with $\overline{\alpha}_S \preceq \overline{\beta}_S$.
Exploiting this assumption and using the same reasoning schema used in (b) we deduce the absurd that $\langle S, \preceq\rangle$ is not stable.

□

Proof of Lemma 4.22. We know that, by definition, $\langle S, \preceq\rangle \sqsubseteq \langle \Pi, Q^+\rangle$. Hence we only have to prove that Q^+ is stable with respect to G. If $\alpha, \beta, \gamma \in \Pi$ are such that $(\alpha, \beta) \in Q^+$ and $\alpha \rightarrow_\exists \gamma$, then from the condition in (1σ) applied to α and γ we have that there exists $\gamma_1' \in \Pi$ such that $(\gamma, \gamma_1') \in Q$ and $\alpha \rightarrow_\forall \gamma_1'$. Let $\beta_1, \ldots, \beta_n$ be such that $(\alpha, \beta_1) \in Q$, $(\beta_i, \beta_{i+1}) \in Q$ and $\beta_n = \beta$. Using the second condition of (2σ) on (α, β_1) and γ_1' we obtain that there exists γ_1'' such that $(\gamma_1', \gamma_1'') \in Q$ and $\beta_1 \rightarrow_\exists \gamma_1''$. Now applying (1σ) to β_1 and γ_1'' we obtain that there exists γ_1''' such that $(\gamma_1'', \gamma_1''') \in Q$ and $\beta_1 \rightarrow_\forall \gamma_1'''$. Similarly by induction on n we obtain a chain of classes of Q of the form $\gamma, \gamma_1', \gamma_1'', \gamma_1''', \ldots, \gamma_n', \gamma_n'', \gamma_n'''$ such that $\beta \rightarrow_\forall \gamma_n'''$. Hence $(\gamma, \gamma_n''') \in Q^+$ and $\beta \rightarrow_\forall \gamma_n'''$, i.e. our thesis. □

Proof of Theorem 4.24. Since P is transitive we have that $\langle S, \preceq\rangle \sqsubseteq \langle \Sigma, P\rangle$. Hence applying Lemma 4.21 we obtain that $\langle S, \preceq\rangle \sqsubseteq \sigma(\langle \Sigma, P\rangle)$. Assume that we have proved that $\langle S, \preceq\rangle \sqsubseteq \sigma^i(\langle \Sigma, P\rangle)$ applying Lemma 4.21 we obtain that $\langle S, \preceq\rangle \sqsubseteq \sigma^{i+1}(\langle \Sigma, P\rangle)$.
Hence we have obtained that for all $i \in \mathbb{N}$ it holds that

$$\langle S, \preceq\rangle \sqsubseteq \sigma^i(\langle \Sigma, P\rangle) \sqsubseteq \langle \Sigma, P\rangle.$$

Moreover from the fact that P is transitive we obtain that

$$\sigma^i(\langle \Sigma, P\rangle)^+ \sqsubseteq \langle \Sigma, P\rangle,$$

where $\sigma^i(\langle\Sigma, P\rangle)^+$ is the transitive closure of the relation in $\sigma^i(\langle\Sigma, P\rangle)$. Hence we have that

$$\langle S, \preceq\rangle \sqsubseteq \sigma^i(\langle\Sigma, P\rangle)^+,$$

so applying Lemma 4.23 we obtain that for all i it holds that $\langle S, \preceq\rangle$ is also the solution of the GCPP over $\sigma^i(\langle\Sigma, P\rangle)$.
Applying Lemma 4.22 to $\sigma^n(\langle\Sigma, P\rangle)$ we obtain $\langle S, \preceq\rangle = \sigma^n(\langle\Sigma, P\rangle)^+$ and we already have that $\langle S, \preceq\rangle \sqsubseteq \sigma^n(\langle\Sigma, P\rangle)$, hence it must be $\langle S, \preceq \rangle = \sigma^n(\langle\Sigma, P\rangle)$. □

Proof of Theorem 4.25. Let $\langle S, \preceq\rangle$ be the solution of the GCPP on G and $\langle\Sigma, P\rangle$. Using the same reasoning-schema of Theorem 4.24 we obtain that $\langle S, \preceq\rangle \sqsubseteq \langle\Sigma_i, \mathsf{P}_{i+1}\rangle \sqsubseteq \langle\Sigma_i, \mathsf{P}_i\rangle$. If we prove that $\langle\Sigma_i, \mathsf{P}_i^+\rangle$ is stable we can conclude by Lemma 4.14 that $\langle S, \preceq\rangle \sqsupseteq \langle\Sigma_i, \mathsf{P}_i^+\rangle$ and hence $\langle S, \preceq\rangle = \langle\Sigma_i, \mathsf{P}_i\rangle$ and $\mathsf{P}_{i+1} = \mathsf{P}_i$.
Let $\alpha, \beta, \gamma \in \Sigma_i$ be such that $(\alpha, \beta) \in \mathsf{P}_i^+$ and $\alpha \rightarrow_\exists \gamma$. From the fact that $(\alpha, \beta) \in \mathsf{P}_i^+$ we have that there exists $(\alpha, \beta_1), \ldots, (\beta_n, \beta) \in \mathsf{P}_i$. Since $\alpha \rightarrow_\exists \gamma$ and $\Sigma_{i+1} = \Sigma_i$, we have that there exists γ' such that $(\gamma, \gamma') \in \mathsf{P}_i$ and $\alpha \rightarrow_\forall \gamma'$. Since P_i is obtained from $\sigma^i(\langle\Sigma, P\rangle)$ we obtain that there exists δ_1' such that $(\gamma', \delta_1') \in \mathsf{P}_i$ and $\beta_1 \rightarrow_\exists \delta_1'$. Using again the fact that $\Sigma_{i+1} = \Sigma_i$ we obtain that there exists δ_1 such that $(\delta_1', \delta_1) \in \mathsf{P}_1$ and $\beta_1 \rightarrow_\forall \delta_1$. By induction on n we arrive to the conclusion that there exists δ such that $(\gamma, \delta) \in \mathsf{P}_i^+$ and $\beta \rightarrow_\forall \delta$. □

Proof of Proposition 4.28. In order to have our thesis we only need to prove that

$$\forall\gamma \in \Pi(\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma \Rightarrow \exists\gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \wedge \beta \rightarrow_\exists^\Sigma (\Pi)\gamma')) \Leftrightarrow$$

$$\forall\gamma \in \Sigma(\alpha \rightarrow_\forall \gamma \Rightarrow \exists\gamma' \in \Sigma((\gamma, \gamma') \in P \wedge \beta \rightarrow_\exists \gamma)).$$

(⇒) Assume, by contradiction, that

$$\forall\gamma \in \Pi(\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma \Rightarrow \exists\gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \wedge \beta \rightarrow_\exists^\Sigma (\Pi)\gamma')) \wedge$$

$$\exists\gamma \in \Sigma(\alpha \rightarrow_\forall \gamma \wedge \forall\gamma' \in \Sigma((\gamma, \gamma') \in P \Rightarrow \neg\beta \rightarrow_\exists \gamma)). \qquad (3)$$

Let $\gamma \in \Sigma$ be the class such that $(\alpha \rightarrow_\forall \gamma \wedge \forall\gamma' \in \Sigma((\gamma, \gamma') \in P \Rightarrow \neg\beta \rightarrow_\exists \gamma))$. As $\alpha \rightarrow_\forall \gamma$ we can find a class of Π, γ^1 such that $\alpha \rightarrow_\exists \gamma^1 \wedge \gamma^1 \subseteq \gamma$.

By construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ we have that $\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma^1$ and hence (by equation 3) that there exists a class in Π, say δ^1, such that $\beta \rightarrow_\exists^\Sigma (\Pi)\delta^1 \wedge (\gamma^1, \delta^1) \in P(\Pi)$.
Let δ be the class of Σ such that $\delta^1 \subseteq \delta$. As $\beta \rightarrow_\exists^\Sigma (\Pi)\delta^1 \wedge (\gamma^1, \delta^1) \in P(\Pi)$ we deduce, by definition of $P(\Pi)$ and by construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$,

that $\beta \rightarrow_\exists \delta \wedge (\gamma, \delta) \in P$ which contradicts our assumption. ($\Leftarrow$) Assume by contradiction that

$$\forall \gamma \in \Sigma(\alpha \rightarrow_\forall \gamma \Rightarrow \exists \gamma' \in \Sigma((\gamma, \gamma') \in P \wedge \beta \rightarrow_\exists \gamma)) \wedge$$

$$\exists \gamma \in \Pi(\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma \wedge \forall \gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \Rightarrow \neg\beta \rightarrow_\exists^\Sigma (\Pi)\gamma')) \quad (4)$$

Let γ^1 be the class of Π such that $\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma^1 \wedge \forall \delta^1 \in \Pi((\gamma^1, \delta^1) \in P(\Pi) \Rightarrow \neg\beta \rightarrow_\exists^\Sigma (\Pi)\delta^1)$. Consider $\gamma \in \Sigma$ such that $\gamma^1 \subseteq \gamma$. By construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ (as $\alpha \rightarrow_\forall^\Sigma (\Pi)\gamma^1$) we have that $\alpha \rightarrow_\forall \gamma$ and hence, by equation (4), that $\exists \delta \in \Sigma$ such that $(\gamma, \delta) \in P \wedge \beta \rightarrow_\exists \delta$. Let $\delta^1 \in \Pi$ be such that $\delta^1 \subseteq \delta$. As $\beta \rightarrow_\exists \delta \wedge (\gamma, \delta) \in P$ we deduce, by definition of $P(\Pi)$ and by construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$, that $\beta \rightarrow_\exists^\Sigma (\Pi)\delta^1 \wedge (\gamma^1, \delta^1) \in P$ which contradicts our assumption. □

Proof of Lemma 4.31. Let $P_1 = \mathbf{New\_HHK}(D, K, \bot)$. Let $(a, b) \notin P_1$, we have to prove that $(a, b) \notin \tau_D(K)$. If $(a, b) \notin K$, then the result is trivial. If $(a, b) \in K$, from $(a, b) \notin P_1$ we have that there exists c such that $a \in pre_2(c), b \in rem(c)$ and $b \in sim(a)$. From $a \in pre_2(c)$ we have that aR_2c. From $b \in rem(c)$, since we are in the case in which $U = \bot$, we have that $b \in T \setminus pre_1(\{e \mid (c, e) \in K\})$ (notice that $rem(c)$ is never modified and when it is build the initial definition of $sim(c)$ is $\{e \mid (c, e) \in K\}$). This means that $\forall e((c, e) \in K \Rightarrow \neg bR_1e)$, i.e. $\forall d(bR_1d \Rightarrow (c, d) \notin K)$. Hence we have obtained aR_2c and $\forall d(bR_1d \Rightarrow (c, d) \notin K)$, which implies that (a, b) is not in $\tau_D(K)$. Now we assume that $(a, b) \notin \tau_D(K)$ and we prove that $(a, b) \notin P_1$. If $(a, b) \notin K$ is trivial. If $(a, b) \in K$, then from the definition of τ_D, we have that there exists c such that aR_2c and $\forall d(bR_1d \Rightarrow (c, d) \notin K)$. This implies that $a \in pre_2(c)$ and $b \in rem(c)$. From the fact that $(a, b) \in K$ we also have in the initialization it holds $b \in sim(a)$. We can assume that c is the first such that $rem(c) \neq \emptyset$ and $b \in rem(c), a \in pre_2(c)$. This means that also $b \in sim(a)$ still holds. Hence, at this step (a, b) is removed from P_1, i.e. the thesis.

Let $P_2 = \mathbf{New\_HHK}(D, K, \top)$. Let $(a, b) \notin P_2$ we have to prove that $(a, b) \notin Fix(\tau_D)(K)$. The case in which $(a, b) \notin K$ is trivial. Let k be the number of extractions performed on $LS = \{c \mid rem(c) \neq \emptyset\}$ at the iteration in which (a, b) is removed from P_2. We proceed by induction on k. If $k = 1$, then c is the first element we extract from LS, hence $rem(c) := T \setminus pre_1(\{e \mid (c, e) \in K\})$, i.e. $rem(c) = \{b \mid \forall d(bR_1d \Rightarrow (c, d) \notin K\}$. Since we remove (a, b) from P_2 this means that $aR_2c, b \in rem(c)$ and $b \in sim(a)$, and using the fact that $rem(c) = \{b \mid \forall d(bR_1d \Rightarrow (c, d) \notin K\}$, this implies that $(a, b) \notin \tau_D(K)$, hence $(a, b) \notin Fix(\tau_D)(K)$. Let us assume we have proved the thesis in the case $k \leq h$, and (a, b) is such that it is removed from P_2 at the $(h+1)$-th

extraction from LS. This means that there exists c such that aR_2c and $b \in rem(c)$. If b was in $rem(c)$ in the initialization then, as in the base case, we can prove that $(a,b) \notin \tau_D(K)$. If b has been added to $rem(c)$ after the initialization, then there exists $h' \leq h$ such that at the h'-th extraction from LS we have the following situation: a pair (c,d) has been removed from P_2, $b \in pre_1(d)$ and $post_1(b) \cap sim(c) = \emptyset$. It is immediate to prove that the following invariant holds:

$$\forall c \in T(sim(c) = \{e \mid (c,e) \in P\}).$$

Hence, we have that the situation at the h'-th can be rewritten as (c,d) has been removed from P_2, $b \in pre_1(d)$ and $\forall d'(bR_1d' \Rightarrow (c,d') \notin P_2)$. By inductive hypothesis we have that $(c,d) \notin Fix(\tau_D)(K)$ and the same for each d' such that bR_1d', i.e. there exists i such that $(c,d) \notin \tau_D^i(K)$ and $\forall d'(bR_1d' \Rightarrow (c,d') \notin \tau_D^i(K))$. We have obtained that aR_2c and $\forall d'(bR_1d' \Rightarrow (c,d') \notin \tau_D^i(K))$, hence $(a,b) \notin \tau_D^{i+1}(K)$. We now prove that if $(a,b) \notin Fix(\tau_D(K))$, then $(a,b) \notin P_2$. If $(a,b) \notin K$, then is trivial. Let k be the number such that $(a,b) \in \tau_D^{k-1}(K)$ and $(a,b) \notin \tau_D^k(K)$. We proceed by induction on k and we prove that $(a,b) \notin \tau_D^k(K)$ implies $(a,b) \notin P_2$. If $k = 1$, then this is equivalent to say that there exists c such that aR_2c and $\forall d(bR_1d \Rightarrow (c,d) \notin K)$. This implies that initially $b \in rem(c)$. Since we are assuming that $(a,b) \in K$ we also have that initially $b \in sim(a)$. Hence, when c is extracted for the first time from LS (since T is finite we are sure that if $sim(c) \neq \emptyset$, then there exists an iteration in which c is extracted from LS), if (a,b) has not yet been removed from P_2 we have that aR_2c, $b \in rem(c)$, and $b \in sim(a)$. Hence, at this point (a,b) is removed from P_2. Let us assume we have proved the thesis in the case $k \leq h$ and we prove the thesis for $k+1$. Our hypothesis is that $(a,b) \in \tau_D^k(K)$ and $(a,b) \notin \tau_D^{k+1}(K)$. This means that there exists c such that aR_2c and $\forall d(bR_1d \Rightarrow (c,d) \notin \tau_D^k(K))$. Applying the inductive hypothesis we obtain that $\forall d(bR_1d \Rightarrow (c,d) \notin P_2)$. When we remove from P_2 the last pair (c,d') such that bR_1d' we have that $b \in prec_1(d')$ and $post_1(b) \cap sim(c) = \emptyset$. This implies that b is added to $rem(c)$. Let us consider the first iteration after the one in which we add b to $rem(c)$ in which c is extracted from LS. We have that aR_2c, $b \in rem(c)$, and, assuming that (a,b) has not yet been removed from P_2, $b \in sim(a)$, hence at this point (a,b) is removed from P_2. □